RUTINAS EN LISP

6

6 Rutinas del Entorno de Diseño 6.1

6.1 Módulo con rutinas de carácter Básico y Genérico que se utiliza en el resto de los módulos 6.1
6.1.1 Rutinas Básicas de CAD 6.2
6.1.2 Rutinas Genéricas de EFCiD 6.19

6.2 Módulo de Generación de Prototipos Estructurales 6.48
6.2.1 Funciones Básicas 6.49
6.2.2 Generación de Celosías planas 6.50
6.2.3 Generación de Celosías tridimensionales 6.58
6.2.4 Generación de Mallas Espaciales 6.63
6.2.5 Generación de Sistemas Estructurales desarrollados sobre Superficies 6.80
6.2.6 Generación de Sistemas Estructurales por volúmenes 6.88
6.2.7 Generación de Vigas y Porticos 6.101
6.2.8 Generación de Forjados Reticulares 6.108

6.3 Módulo para describir Características Geométricas y Mecánicas de los elementos estructurales 6.113

6.4 Módulo para obtener las propiedades Mecánicas de una sección y la distribución de tensiones normales 6.127

6.5 Módulo de aplicación de Vínculos con el contorno y descripción de Ligaduras entre barras 6.142
6.5.1 Vínculos de tipo constructivo 6.142
6.5.2 Vínculo de tipo ideal 6.148

6.6 Módulo de aplicación de Cargas 6.152
6.6.1 Funciones para aplicar Cargas directamente 6.152
6.6.2 Funciones para aplicar Cargas a través de los forjados 6.164

6.7 Módulo de Cálculo y Trazado de Forjados 6.167

6 Rutinas del Entorno de Diseño

Con el objeto de facilitar la personalización del uso del entorno de Diseño del programa EFCiD se incluye el código fuente de las rutinas que lo componen y que están escritas en el lenguaje Visual LISP.

Todas las rutinas que a continuación se presentan ellas forman parte del programa de Cálculo y Diseño de estructuras: EFCiD v 7.20 (2003). Este es un programa Registrado cuyos derechos pertenecen a la Universidad Politécnica de Valencia.

Copyright: Universidad Politécnica de Valencia
Departamento: M.M.C. y Teoría de Estructuras
Grupo I+D+I: Grupo de Calculo y Diseño Estructural en Edificación- CiD
Autores: AGUSTIN PEREZ GARCIA & ADOLFO ALONSO DURA
Contacto: aperezg@mes.upv.es aalonsod@mes.upv.es
Domicilio: Camino de Vera s/n 46021 VALENCIA

Teléfono: 96 3877671
Fax: 96 3879679

La distribución de las rutinas, por cualquier medio, queda reservada a los autores y al titular del Copyright. No obstante, pueden ser utilizadas por los miembros de la comunidad académica universitaria siempre que se citen las fuentes y los autores. Aunque las rutinas han sido comprobadas exhaustivamente, los autores no aceptan responsabilidad alguna respecto del uso de las mismas por terceras partes.

6.1 Módulo con rutinas de carácter Básico y Genérico que se utiliza en el resto de los módulos

Este conjunto de rutinas se clasifican en:

- funciones de uso genérico en cualquiera de los entornos de diseño utilizados (AutoCAD o IntelliCAD) que permiten gestionar entidades gráficas.
- funciones específicas del Entorno de Diseño de EFCiD pero utilizadas por varios de sus módulos.

6.1.1 Rutinas Básicas de CAD

```
; ************ FUNCIONES CON UTILIDADES BASICAS PARA GESTIONAR OTRAS RUTINAS


; C:H        ACTIVA EL RESALTE DE LAS ENTIDADES SELECCIONADAS
; noecho     DESACTIVA EL ECO DE LOS COMANDOS
; diasi      DESACTIVA LA PRESENTACION DE CUADROS DE DIALOGO
; diano      ACTIVA LA PRESENTACION DE CUADROS DE DIALOGO
; CreaTl     CREA UN NUEVO TIPO DE LINEA PARA ATRIBUIR PROPIEDADES ESTRUCTURALES
; CargaTl    CARGA UN TIPO DE LINEA PARA ASIGNAR PROPIEDADES ESTRUCTURALES
; puntint    CALCULA UN PUNTO INTERMEDIO ENTRE OTROS DOS POR UN FACTOR
; pmig       PUNTO MEDIO ENTRE LOS PUNTOS P1 Y P2
; pfrac      PUNTO INTERMEDIO EN UNA FRACCION (f partes iguales) ENTRE  pi Y pf
; l3p        DEVUELVE la LISTA (Pini Pfin Pmed) de una entidad
; contvpol   FORMA UNA LISTA CON LOS VERTICES DE UNA POLILINEA LWPOL
; contv3dpol FORMA UNA LISTA CON LOS VERTICES DE UNA 3DPOLILINEA
; long_pol   FORMA UNA LISTA CON LOS VERTICES DE UNA POLILINEA 3D
; exlist     PRODUCTO DE UN ESCALAR POR UNA LISTA
; altp       DEVUELVE UN PUNTO CON UNA ALTURA H
; provec     PRODUCTO VECTORIAL
; prod3x1    PRODUCTO DE UNA MATRIZ 3X3 POR UN VECTOR
; mrot       FORMA LA MATRIZ DE ROTACION DE EJES LOCALES A GLOBALES
; sentit     DETECTA SENTIDO HORARIO DEL RECORRIDO DE LOS VERTICES DE UNA lwpol
; r_non      REFERENCIA A ENTIDADES
; r_cer
; r_per
; r_int
; r_fin
; r_med
; r_pto
; r_fmi
; cazapuntos
; cprev      GESTION DEL ENTORNO DE ENTRADA Y SALIDA DE FUNCIONES
; c:rv
; avisoUNDO
; cpcap
; pgcap
; cposm
; pgosm
; cpscp
; pgscp
; scpu
; cpvista
; pgvista
; inscapa    Insertar un bloque en una capa determinada
; ir_a_capa  Establece una capa como actual. Reutiliza y activa si es necesario
; C:AA       Desactiva la capa de una determinada entidad
; C:AAA      Desactiva la capa de una determinada subentidad dentro de un bloque
; C:BB       Bloquea la capa de una determinada entidad
; C:DD       Desbloquea la capa de una determinada entidad
; C:CA       Establece como actual la capa de una determinada entidad
; C:FF       Inutiliza la capa de una determinada entidad
; C:VV       Vacia una capa borrando todas sus entidades
; dellerr
; indice     Busca la posicion que ocupa un elemento dentro una lista
; posicion   Busca la posicion que deberia ocupar un elemento para quedar ordenado
; getconj    FUNCIONES BASICAS
; copiaconj
; capaconj
```

```
; getent
; getsubent
; getrotul
; getsubcapa
; getcolor
; getlinea
; copiacapa  Copia todas las entidades de una capa a otra capa
; C:CC       Copia entidades de distintas capas a una determinada capa
; C:CCC      Selecciona las capas origen y destino y copia las entidades de una a
otra
; C:TT       Traslada entidades de distintas capas a una determinada capa
; C:TTT      Traslada todas las entidades de una capa a otra capa
; C:E        Borra una entidad o un grupo de entidades


(prompt "Cargando las utilidades  BASICAS  \n")


;*****************************************************************************
;* * * INICIALIZACIONES
;*****************************************************************************


(setvar "MIRRTEXT" 0)


;*****************************************************************************
;* * *   FUNCIONES BASICAS
;*****************************************************************************


;*****************************************************************************
;* * *   ACTIVA EL RESALTE DE LAS ENTIDADES SELECCIONADAS
;*****************************************************************************

(defun C:H ()

  (setvar "HIGHLIGHT" 1)
)


;*****************************************************************************
;* * *   DESACTIVA EL ECO DE LOS COMANDOS
;*****************************************************************************

(defun noecho ()

  (setvar "CMDECHO" 0)
)


;*****************************************************************************
;* * *   DESACTIVA LA PRESENTACION DE CUADROS DE DIALOGO
;*****************************************************************************

(defun diasi ()

  (setvar "ATTDIA" 1)
)


;*****************************************************************************
;* * *   ACTIVA LA PRESENTACION DE CUADROS DE DIALOGO
;*****************************************************************************

(defun diano ()
```

```
  (setvar "ATTDIA" 0)
)


;*****************************************************************************
;* * * CREA UN NUEVO TIPO DE LINEA PARA ATRIBUIR PROPIEDADES ESTRUCTURALES
;*****************************************************************************

(defun CreaTl (tpl)
  (command "_LINETYPE"          "_C"                tpl
           "c:/cid/cad/st.lin" "Define tipo de elemento"
           "12,-0.1"           ""
          )
)


;*****************************************************************************
;* * * CARGA UN TIPO DE LINEA PARA ASIGNAR PROPIEDADES ESTRUCTURALES
;*****************************************************************************

(defun CargaTl (tpl)
  (command "_LINETYPE" "_L" tpl "c:/cid/cad/st.lin" "")
)


;*****************************************************************************
;* * * CALCULA UN PUNTO INTERMEDIO ENTRE OTROS DOS POR UN FACTOR
;*****************************************************************************

(defun puntint (pin pf fact  / pm x y z)
            (setq x (+ (* (- (car pf) (car pin)) fact) (car pin))
                  y (+ (* (- (cadr pf) (cadr pin)) fact) (cadr pin))
                  z (+ (* (- (caddr pf) (caddr pin)) fact) (caddr pin))
                  pm (list x y z))
)


;*****************************************************************************
;* * * PUNTO MEDIO ENTRE LOS PUNTOS P1 Y P2
;*****************************************************************************

(defun pmig (pin pf / pm  x y z)

            (setq x  (/ (+ (car pin) (car pf)) 2)
                  y  (/ (+ (cadr pin) (cadr pf)) 2)
                  z  (/ (+ (caddr pin) (caddr pf)) 2)
                  pm (list x y z)
            )
)


;*****************************************************************************
;* * *  PUNTO INTERMEDIO EN UNA FRACCION (f partes iguales) ENTRE  pi Y pf
;*****************************************************************************


(defun pfrac (pin pf f / pp xn yn zn x y z)

       (setq xn (/ (- (car   pf) (car   pin)) f)
             yn (/ (- (cadr  pf) (cadr  pin)) f)
             zn (/ (- (caddr pf) (caddr pin)) f)
             x  (+ xn (car   pin))
             y  (+ yn (cadr  pin))
             z  (+ zn (caddr pin))
             pp (list x y z)
```

```
      )
)


;*****************************************************************************
;* * *  DEVUELVE la LISTA (Pini Pfin Pmed) de una entidad
;
;       Pini     punto inicial
;       Pfin     punto final
;       Pmed     punto medio
;
;*****************************************************************************

(defun l3p (ent / n1 n0 p1 p2 pm lp an1 an2 c cc r pp1 pp2 pp3 cero pd)

  (setq cero (list 1.0 1.0 1.0)
        n0   (car ent)
        n1   (entget n0)
        pd   (cdr ent)
  )
  (if (= "LINE" (cdr (assoc 0 n1)))
    (setq p1 (cdr (assoc 10 n1))
          p2 (cdr (assoc 11 n1))
          pm (pmig p1 p2)
    )
  )
  (if (= "ARC" (cdr (assoc 0 n1)))
    (progn
      (SCPObjeto ent)
      (setq c    (list 0.0 0.0 0.0)
            an1  (cdr (assoc 50 n1))
            an2  (cdr (assoc 51 n1))
            r    (cdr (assoc 40 n1))
            an2  (- an2 an1)
            an1  0.0
            pp1  (polar c an1 r)
            pp2  (polar c an2 r)
            pp3  (polar c (* 0.5 (+ an2 an1)) r)
            p1   (trans pp1 1 0)
            p2   (trans pp2 1 0)
            pm   (trans pp3 1 0)
      )
      (scpu)
    )
  )
  (if (= "LWPOLYLINE" (cdr (assoc 0 n1)))
    (setq an1 (contvpol n0)
          an2 (nth 0 an1)
          pp2 (/ an2 2)
          p1  (nth 1 an1)
          p2  (nth an2 an1)
          pm  (nth pp2 an1)
    )
  )
  (if (= "POLYLINE" (cdr (assoc 0 n1)))
    (setq an1 (contv3dpol n0)
          an2 (nth 0 an1)
          pp2 (/ an2 2)
          p1  (nth 1 an1)
          p2  (nth an2 an1)
          pm  (nth pp2 an1)
    )
  )
  (setq lp (list p1 p2 pm))
)
```

```
;*****************************************************************************
;* * *  FORMA UNA LISTA CON LOS VERTICES DE UNA POLILINEA LWPOL
;*****************************************************************************

(defun contvpol (ent / e0 nv1 ne e1 nv p1 p2 lp lll z)

  (setq  e0   (entget ent)
         nv1  1
         ne   2
         nv   (cdr (assoc 90 e0))
         lll  '(1 1 1)
         z    (cdr (assoc 38 e0))
  )
  (while (>= nv nv1)
    (setq e1   (assoc 10 e0)
          p1   (cdr e1)
          p1   (reverse p1)
          p1   (cons z p1)
          p1   (reverse p1)
          p2   (trans p1 ent 0)
          lp   (cons p2 lp)
          nv1  (1+ nv1)
          e0   (subst lll e1 e0)
    )
  )
  (setq  lp (reverse lp)
         lp (cons nv lp)                  ; lp (nvert v1 v2 v3... vn)
  )
)


;*****************************************************************************
;* * *  FORMA UNA LISTA CON LOS VERTICES DE UNA 3DPOLILINEA
;*****************************************************************************

(defun contv3dpol (ent / e0 nv1 e1 nv nv1 p1 lp vv ent1 ent2)

  (setq  e0   (entget ent)

         nv1  0
         ent1 ent
         nv   1
  )
  (while (> nv 0)
    (setq ent2 (entnext ent1)
          e0   (entget ent2)
          vv   (cdr (assoc 0 e0))
    )
    (if (= vv "VERTEX")
      (setq e1    (assoc 10 e0)
            p1    (cdr e1)
            ent1 ent2
            lp    (cons p1 lp)

            nv1   (1+ nv1)
      )
      (setq nv 0)
    )
  )
  (setq  lp (reverse lp)
         lp (cons nv1 lp)          ; lp (nvert v1 v2 v3... vn)
  )
)
```

```
;*****************************************************************************
;* * *  FORMA UNA LISTA CON LOS VERTICES DE UNA POLILINEA 3D
;
; lp             (nvert   long.tot   v1  v2  v3...  vn)
;
;*****************************************************************************

(defun long_pol (noment / nom0 nom1 e0 nv1 ne e1 nv p1 lp lon p0)

  (setq  e0   (entget noment)
         nv   0
         nv1  1
         lon  0.0
         nom0 (entnext noment)
         e0   (entget nom0)
  )
  (while (= nv 0)
    (setq p1 (cdr (assoc 10 e0)))
    (if (< 1 nv1)
      (setq lon (+ lon (distance p0 p1)))
    )
    (setq lp   (cons p1 lp)
          nv1  (1+ nv1)
          nom1 nom0
          nom0 (entnext nom1)
          e0   (entget nom0)
    )
    (if (= "SEQEND" (cdr (assoc 0 e0)))
      (setq nv 1)
    )
    (setq p0 p1)
    (if (< 15 nv1)
      (setq nv 1)
    )
  )
  (setq  lp (reverse lp)
         lp (cons lon lp)
         lp (cons nv1 lp)
  )
)


;*****************************************************************************
;* * * PRODUCTO DE UN ESCALAR POR UNA LISTA
;*****************************************************************************

(defun exlist (l esc / lista e c)

  (setq lista (reverse l))
  (foreach e lista (setq c (cons (* esc e) c)))
)


;*****************************************************************************
;* * *    DEVUELVE UN PUNTO CON UNA ALTURA H
;*****************************************************************************

(defun altp (pin hi / h1 p)

  (setq  h1 (list (nth 0 pin) (nth 1 pin) (+ (nth 2 pin) hi))
         p  h1
  )
)


;*****************************************************************************
```

```
;* * *    PRODUCTO VECTORIAL
;
;        devuelve la lista lp ( p1 p2 p3)
;*****************************************************************************
(defun provec (v1                       ; Vector en formato lista (x1 y1 z1)
               v2                       ; Vector en formato lista (x2 y2 z2)
               / x1 y1 z1 x2 y2  z2 x y z lp)

  (setq x1 (nth 0 v1)
        y1 (nth 1 v1)
        z1 (nth 2 v1)
  )
  (setq x2 (nth 0 v2)
        y2 (nth 1 v2)
        z2 (nth 2 v2)
  )
  (setq x (- (* y1 z2) (* z1 y2))
        y (- (* z1 x2) (* x1 z2))
        z (- (* x1 y2) (* y1 x2))
  )
  (setq  lp (list x y z)
  )
)


;*****************************************************************************
;* * *  PRODUCTO DE UNA MATRIZ 3X3 POR UN VECTOR
;
;       devuelve una lista con las tres componentes del vector
;*****************************************************************************

(defun prod3x1 (mt                      ; matriz en forma de lista fila x columna
                                        ; ij mt (11 12 13 21 22 23 31 32 33)
                vv                      ; vector que multiplica a la matriz
                / s1 s2  s3 ix)


  (setq  s1 (* (nth 0 mt) (nth 0 vv))
         s1 (+ s1 (* (nth 1 mt) (nth 1 vv)))
         s1 (+ s1 (* (nth 2 mt) (nth 2 vv)))
  )
  (setq  s2 (* (nth 3 mt) (nth 0 vv))
         s2 (+ s2 (* (nth 4 mt) (nth 1 vv)))
         s2 (+ s2 (* (nth 5 mt) (nth 2 vv)))
  )
  (setq  s3 (* (nth 6 mt) (nth 0 vv))
         s3 (+ s3 (* (nth 7 mt) (nth 1 vv)))
         s3 (+ s3 (* (nth 8 mt) (nth 2 vv)))
  )
  (setq ix (list s1 s2 s3))
)


;*****************************************************************************
;* * * FORMA LA MATRIZ DE ROTACION DE EJES LOCALES A GLOBALES
;      dados los cosenos directores dx dy dz del eje z del sistema local
;*****************************************************************************

(defun mrot (dx                         ; coseno director respecto del eje OX
             dy                         ; coseno director respecto del eje OY
             dz                         ; coseno director respecto del eje OZ
             / x y z v1  v2 v3 v4 v5 v6 v7 v8 v9  l1 l2 l3)

  (setq l1 (list dx dy dz))
  (if (= (abs dz) 1)
```

```
    (setq l2 (list 0 1 0))
    (setq l2 (list 0 0 1))
  )
  (setq l3 (provec l1 l2))
)


;*****************************************************************************
;* * *    DETECTA SENTIDO HORARIO DEL RECORRIDO DE LOS VERTICES DE UNA lwpol
;*****************************************************************************

(defun sentit (lp / x y z s v v1 v2)

  (setq x  (- (nth 0 (nth 2 lp)) (nth 0 (nth 1 lp)))
        y  (- (nth 1 (nth 2 lp)) (nth 1 (nth 1 lp)))
        v1 (list x y 0)
        x  (- (nth 0 (nth 3 lp)) (nth 0 (nth 1 lp)))
        y  (- (nth 1 (nth 3 lp)) (nth 1 (nth 1 lp)))
        v2 (list x y 0)
        v  (prodvec v1 v2)
        z  (nth 2 v)
        s  1.0
  )
  (if (> 0 z)
    (setq s -1.0)
  )
  (setq x s)
)


;*****************************************************************************
;* * * REFERENCIA A ENTIDADES
;*****************************************************************************


(defun r_non ()                         ; Ninguna referencia

  (setvar "osmode" 0)
)

(defun r_cer ()                         ; Punto mas cercano

  (r_non)
  (setvar "osmode" 512)
)

(defun r_per ()                         ; Perpendicular

  (r_non)
  (setvar "osmode" 128)
)

(defun r_int ()                         ; Intersección

  (r_non)
  (setvar "osmode" 32)
)

(defun r_fin ()                         ; Punto final

  (r_non)
  (setvar "osmode" 1)
)
```

```
(defun r_med ()                         ; Punto medio

  (r_non)
  (setvar "osmode" 2)
)

(defun r_pto ()                         ; Entidad tipo punto

  (r_non)
  (setvar "osmode" 8)
)

(defun r_fmi ()                         ; Captura el Punto final, Medio o
                                        ; Interseccion

  (r_non)
  (setvar "osmode" 1059)
)

(defun cazapuntos ()                    ; Captura el punto Final, Medio,
                                        ; Perpendicular o Intersecci¢n que
                                        ; se encuentra mas cercano al cursor.
                                        ; Funciona como un comutador
  (if (= (getvar "OSMODE") 0)
    (setvar "OSMODE" 1067)
    (setvar "OSMODE" 0)
  )
  (setq kk nil)
)


;*****************************************************************************
;* * * GESTION DEL ENTORNO DE ENTRADA Y SALIDA DE FUNCIONES
;*****************************************************************************


(defun cprev ()                         ; Coloca una marca para deshacer con rv

  (command "_UNDO" "_M")
)



(defun c:rv ()                          ; Deshace con rv hasta la marca colocada
                                        ; anteriormente
  (command "_UNDO" "_B")
)


(defun avisoUNDO ()

  (prompt "Si los resultados obtenidos no son los esperados revoque con  RV ")
)


(defun cpcap ()                         ; Obtiene la capa actual y la memoriza en
                                        ; la variable lyr

  (setq  lyr (getvar "CLAYER")
  )
)


(defun pgcap ()                         ; Restituye la capa memorizada en la
                                        ; variable lyr
```

```
  (setvar "CLAYER" lyr)
)


(defun cposm ()                         ; Obtiene el refent activo y lo memoriza
                                        ; en la variable osm

  (setq  osm (getvar "OSMODE")
  )
)


(defun pgosm ()                         ; Restituye el refent memorizado en la
                                        ; variable osm
  (setvar "OSMODE" osm)
)


(defun cpscp ()                         ; Guarda la información de un determinado
                                        ; SCP como SCPorigen

  (command "_UCS" "_D" "SCPorigen")
  (command "_UCS" "_S" "SCPorigen")
)


(defun pgscp ()                         ; Restituye el SCP al memorizado
                                        ; en SCPorigen

  (command "_UCS" "_R" "SCPorigen")
)


(defun scpu ()                          ; Restituye el SCP Universal

  (command "_UCS" "")
)


(defun cpvista ()                       ; Guarda la información de una
                                        ; determinada VISTA como VISTAorigen

  (command "_VIEW" "_D" "VISTAorigen")
  (command "_VIEW" "_S" "VISTAorigen")
)


(defun pgvista ()                       ; Restituye la VISTA a la memorizada en
                                        ; VISTAorigen

  (command "_VIEW" "_R" "VISTAorigen")
)


;*****************************************************************************
;* * * GESTION DE BLOQUES
;*****************************************************************************


;*****************************************************************************
;* * * Insertar un bloque en una capa determinada
;*****************************************************************************

(defun inscapa (blk                     ; Nombre del bloque
                cap                     ; Nombre de la capa
                fx                      ; Factor de escala a la que se inserta
```

```
              )
  (noecho)
  (ir_a_capa cap)
  (if (= nil fx)
    (setq fx 1.0)
  )
  (command "_INSERT" blk "esc" fx)
)


;*****************************************************************************
;* * * GESTION DE CAPAS
;*****************************************************************************

;* * *            FUNCIONES BASICAS


;*****************************************************************************
;* * * Establece una capa como actual. La reutiliza y la activa si es necesario
;*****************************************************************************

(defun ir_a_capa (cap                     ; Nombre de la capa
                  / act  pl tip c)

  (setq  act (getvar "CLAYER")
         pl  (substr act 1 1)
         tip (substr act 2 3)
  )
  (if (or (/= tip cap) (/= pl planta))
    (progn (setq c (strcat planta cap))
           (command "_LAYER" "_T" c "ACT" c "_S" c "")
    )
  )
)


;*****************************************************************************
;* * * Desactiva la capa de una determinada entidad
;*****************************************************************************
;(defun C:AA
;   (
;     /
;     c
;   )
;   (setq c (getcapa "\nSeleccione la entidad cuya capa quiere APAGAR "))
;   (if (= c (getvar "CLAYER"))

;       (prompt "\nSe ha seleccionado la capa actual\n")
;       (command "CAPA" "DES" c ""))
;)



;*****************************************************************************
;* * * Desactiva la capa de una determinada subentidad dentro de un bloque
;*****************************************************************************

(defun C:AAA (/ c)

  (setq c (getsubcapa "\nSeleccione el atributo que quiere APAGAR "))
  (if (= c (getvar "CLAYER"))
    (prompt "\nSe ha seleccionado la capa actual\n")
    (command "_LAYER" "DES" c "")
  )
)
```

```
;*****************************************************************************
;* * * Bloquea la capa de una determinada entidad
;*****************************************************************************
;(defun C:BB
;   (
;     /
;     c
;   )
;   (setq c (getcapa "\nSeleccione la entidad cuya capa quiere BLOQUEAR "))
;   (if (= c (getvar "CLAYER"))
;       (prompt "\nSe ha seleccionado la capa actual\n")
;       (command "CAPA" "Bloquear" c ""))
;)


;*****************************************************************************
;* * * Desbloquea la capa de una determinada entidad
;*****************************************************************************
;(defun C:DD
;   (
;     /
;     c
;   )
;   (setq c (getcapa "\nSeleccione la entidad cuya capa quiere DesBLOQUEAR "))
;   (if (= c (getvar "CLAYER"))
;       (prompt "\nSe ha seleccionado la capa actual\n")
;       (command "CAPA" "Desbloquear" c ""))
;)


;*****************************************************************************
;* * * Establece como actual la capa de una determinada entidad
;*****************************************************************************

(defun C:CA (/ c)

  (setq
    c (getcapa
        "\nSeleccione una entidad de la capa donde quiere DIBUJAR  "
      )
  )
  (if (= c nil)
    (setq c (getstring "\nNombre de la capa:  "))
  )
  (command "_LAYER" "_S" c "")
)


;*****************************************************************************
;* * * Inutiliza la capa de una determinada entidad
;*****************************************************************************

(defun C:FF (/ c)

  (setq  c (getcapa
             "\nSeleccione la entidad cuya capa quiere INUTILIZAR "
           )
  )
  (if (= c (getvar "CLAYER"))
    (prompt "\nSe ha seleccionado la capa actual\n")
    (command "_LAYER" "_F" c "")
  )
```

```
)


;*****************************************************************************
;* * * Vacia una capa borrando todas sus entidades
;*****************************************************************************

(defun C:VV (/ olderr ocmd L S)

  (setq  olderr   *error*
         *error*  dellerr
  )
  (setq ocmd (getvar "CMDECHO"))
  (noecho)
  (setq  L (strcase
             (getcapa
               "\nSeleccione una entidad de la capa quiere VACIAR  "
             )
           )
  )
  (setq S (ssget "X" (list (cons 8 L))))
  (if S
    (command "_ERASE" S "")
    (princ "La capa no contiene entidades.")
  )
  (setq S nil)
  (setvar "CMDECHO" ocmd)
  (setq *error* olderr)
  (princ)
)


(defun dellerr (s)                      ; Rutina de control de errores

  (if (/= s "Function cancelada")
    (princ (strcat "\nError: " s))
  )
  (setq S nil)
  (setvar "CMDECHO" ocmd)
  (setq *error* olderr)
  (princ)
)


;*****************************************************************************
;* * * GESTION DE LISTAS
;*****************************************************************************



;*****************************************************************************
;* * * Busca la posicion que ocupa un elemento dentro una lista
;*****************************************************************************

(defun indice (a                ; Elemento a posicionar en el interior
                                ;       de la lista.
               l                ; Lista a investigar
              )

  (if (member a l)
    (- (length l) (length (member a l)))
  )
)


;*****************************************************************************
```

```
;* * * Busca la posicion que deberia ocupar un elemento para quedar ordenado
;      por su magnitud entre los elementos de una lista
;*****************************************************************************
;
(defun posicion
                 (a                       ; Elemento a posicionar en el interior
                                          ; de la lista.
                  l                       ; Lista a investigar
                  / i f p c lc)

  (setq i  (car l)
        f  (last l)
        lc l
  )
  (if (> f i)
    (setq c "crece")
  )
  (setq suelo nil
        techo nil
  )
  (if (or (and (>= a i) (<= a f)) (and (<= a i) (>= a f)))
    (if (/= (member a l) nil)
      (setq p      (indice a l)
            techo p
            suelo p
      )
      (if (= c "crece")
        (progn (while (> a (car lc)) (setq lc (cdr lc)))
               (setq techo (indice (car lc) l)
                     suelo (- techo 1)
               )
        )
        (progn (while (< a (car lc)) (setq lc (cdr lc)))
               (setq suelo (indice (car lc) l)
                     techo (- suelo 1)
               )
        )
      )
    )
  )
)


;*****************************************************************************
;* * * EDICION DE ENTIDADES
;*****************************************************************************
;
;* * *                    FUNCIONES BASICAS
;
;*****************************************************************************

(defun getconj (msg)                      ; Selecciona un conjunto de entidades

  (prompt msg)
  (while (not (setq conj (ssget))))
)


;*****************************************************************************

(defun copiaconj (conj)                   ; Duplica un conjunto de entidades

  (if (/= nil conj)
    (command "_COPY" conj "" "0,0" "0,0")
  )
)
```

```
;*****************************************************************************

(defun capaconj  (conj l)         ; Cambia la capa de un conjunto de
                                  ; entidades

  (if (/= nil conj)
    (command "_CHANGE" conj "" "_P" "_LA" l "")
  )
)


;*****************************************************************************

(defun getent (msg)               ; Selecciona una entidad

  (car (entsel msg))
)


;*****************************************************************************

(defun getsubent (msg)            ; Selecciona una subentidad de un bloque

  (car (nentsel msg))
)


;*****************************************************************************

(defun getrotul  (msg / ent)      ; Detecta el rótulo de una entidad

  (setq ent (getent msg))
  (if (/= nil ent)
    (cdr (assoc 5 (entget ent)))
  )
)


;*****************************************************************************

(defun getsubcapa (msg / ent)     ; Detecta la capa en la que se encuentra
                                  ; una subentidad

  (setq ent (getsubent msg))
  (if (/= nil ent)
    (cdr (assoc 8 (entget ent)))
  )
)


;*****************************************************************************

(defun getcolor  (msg / ent col l c)      ; Detecta el color de una entidad o
                                          ; subentidad

  (setq ent (getsubent msg))
  (if (/= nil ent)
    (progn (setq ent (entget ent)
                 col (cdr (assoc 62 ent))
                 l   (cdr (assoc 8 ent))
           )
           (if (= nil col)
             (setq c (cdr (assoc 62 (tblsearch "layer" l))))
             (setq c col)
```

```
          )
    )
  )
)

;*****************************************************************************
(defun getlinea  (msg / ent tl l c)       ; Detecta el tipo de línea de una
                                          ; entidad o subentidad

  (setq ent (getsubent msg))
  (if (/= nil ent)
    (progn (setq ent (entget ent)
                 tl  (cdr (assoc 6 ent))
                 l   (cdr (assoc 8 ent))
           )
           (if (= nil tl)
             (setq c (cdr (assoc 6 (tblsearch "layer" l))))
             (setq c tl)
           )
    )
  )
)

;*****************************************************************************
;* * * Copia todas las entidades de una capa a otra capa
;*****************************************************************************

(defun copiacapa (l                       ; Capa origen
                  nl                      ; Capa destino
                  / conj c)

  (setq  conj (ssget "X" (list (cons 8 l)))
         c    (ssget "X" (list (cons 8 nl)))
  )
  (if (= nil c)
    (progn (copiaconj conj) (capaconj conj nl))
    (progn (prompt (strcat "\nLa capa " nl " no est  vacia."))
           (setq ? (getstring "\n¨Se copian las entidades? S/N"))
           (if (or (= ? "S") (= ? "s"))
             (progn (copiaconj conj)
                    (capaconj conj nl)
             )
           )
    )
  )
)

;*****************************************************************************
;* * * Copia entidades de distintas capas a una determinada capa
;*****************************************************************************

(defun C:CC (/ conj l)


  (noecho)
  (getconj "\nSeleccione entidades a copiar a otra capa: ")
  (setq  l (getcapa
             "\nSeleccione una entidad de la capa a la que se copian: "
           )
  )
  (if (= l nil)
    (setq l (getstring "\nNombre de la capa:  "))
```

```
  )
  (copiaconj conj)
  (capaconj conj l)
)


;*****************************************************************************
;* * * Selecciona las capas origen y destino y copia las entidades de una a otra
;*****************************************************************************

(defun C:CCC (
              /
              l                         ; Capa origen
              nl                ; Capa destino
             )

  (setq l (getcapa "\nSeleccione una entidad de la capa origen: "))
  (if (= l nil)
    (setq l (getstring "\nNombre de la capa origen: "))
  )
  (setq nl (getcapa "\nSeleccione una entidad de la capa destino: "))
  (if (= nl nil)
    (setq nl (getstring "\nNombre de la capa destino: "))
  )
  (copiacapa l nl)
)


;*****************************************************************************
;* * * Traslada entidades de distintas capas a una determinada capa
;*****************************************************************************

(defun C:TT (/ conj l)

  (noecho)
  (getconj "\nSeleccione entidades a cambiar de capa: ")
  (setq
    l (getcapa
        "\nSeleccione una entidad de la capa a la que se trasladan:"
      )
  )
  (if (= l nil)
    (setq l (getstring "\nNombre de la capa:  "))
  )
  (capaconj conj l)
)


;*****************************************************************************
;* * * Traslada todas las entidades de una capa a otra capa
;*****************************************************************************

(defun C:TTT (
              /
              l                         ; Capa origen
              nl                ; Capa destino
             )

  (setq l (getcapa "\nSeleccione una entidad de la capa origen: "))
  (if (= l nil)
    (setq l (getstring "\nNombre de la capa origen: "))
  )
  (setq nl (getcapa "\nSeleccione una entidad de la capa destino: "))
  (if (= nl nil)
    (setq nl (getstring "\nNombre de la capa destino: "))
  )
```

```
  (setq conj (ssget "X" (list (cons 8 l))))
  (capaconj conj nl)
)

;*****************************************************************************
;* * * Borra una entidad o un grupo de entidades
;*****************************************************************************

;(defun C:E ()
;
;  (command "BORRA" "di" "auto")
;)
```

6.1.2 Rutinas Genéricas de EFCiD

```
; ************ FUNCIONES DE CARACTER GENERAL PARA GESTIONAR EL PROGRAMA EFCiD


; C:QQ       DEMANDA DE INFORMACION
; C:SS       INSERCION DE DATOS

; C:DXF      EXPORTA A UN FICHERO EN FORMATO  dxf  LAS ENTIDADES SELECCIONADAS
; C:ZZ       SITUAR EL SCP EN EL PLANO PERPENDICULAR A UNA RECTA DADA
; nomforj    CREA LAS CAPAS PARA UBICAR LOS ELEMENTOS DE UN PORTICO ESPACIAL
; capasstr
; tl->props  MUESTRA LAS CARACTERISTICAS DE LA SECCION DE UNA BARRA
; props->tl  CONSTRUYE EL NOMBRE DEL TIPO DE LINEA CORESPONDIENTE A PATACT
; actualiza  ACTUALIZACION DEL BLOQUE PATACT
; versec     SELECCIONA BARRAS Y DIBUJA EL BLOQUE CORRESPONDIENTE A SU SECCION
; escipn     SELECCIONA LOS FACTORES DE ESCALA DE UN IPN
; escipe     SELECCIONA LOS FACTORES DE ESCALA DE UN IPE
; eschea     SELECCIONA LOS FACTORES DE ESCALA DE UN HEA
; escheb     SELECCIONA LOS FACTORES DE ESCALA DE UN HEB
; eschem     SELECCIONA LOS FACTORES DE ESCALA DE UN HEM
; dibsec     SELECCIONA EL BLOQUE Y ESCALA DE UNA SECCION DADA Y LO DIBUJA
; dv3dc      DIBUJA EL VOLUMEN DE UN ELEMENTO FINITO SUPERFICIAL CON ESPESOR


(prompt "Cargando las utilidades de GENERALES \n")


;*****************************************************************************
;* * * DEMANDA DE INFORMACION
;*****************************************************************************

(defun C:QQ (/ conj ent n0 n1 n2 v0 v1 v2 x1 y1 z1 x2 y2 z2)


  (setq ent (entsel "\nSeleccione la entidad:    "))

  (while (/= nil ent)

    (setq n0 (car ent)
          n1 (entget n0)
```

```
  )

  (if (= "INSERT" (cdr (assoc 0 n1)))
    (command "_DDATTE" n0)
  )

  (if (= "LINE" (cdr (assoc 0 n1)))
    (tl->props n1 "QQ")
  )

  (if (= "3DFACE" (cdr (assoc 0 n1)))
    (tl->props n1 "QQ")
  )

  (if (= "LWPOLYLINE" (cdr (assoc 0 n1)))
    (progn (setq v1 (member (assoc 10 n1) n1)
                 x1 (cdr (assoc 10 v1))
                 v1 (cdr v1)
                 v1 (member (assoc 10 v1) v1)
                 v2 (cdr v1)
                 x2 (cdr (assoc 10 v2))
                 v0 (distance x1 x2)
                 v0 (abs (/ v0 ef))
           )
           (princ "\nEntidad seleccionada -->  Carga de ")
           (princ v0)
           (princ " Toneladas")
           (terpri)
    )
  )

  (if (= "POLYLINE" (cdr (assoc 0 n1)))
    (if (= (cdr (assoc 6 n1)) "CARGASUP")
      (progn (setq v1 (entget (entnext n0))
                   v2 (entget (entnext (entnext (entnext n0))))
                   x1 (cdr (assoc 10 v1))
                   x2 (cdr (assoc 10 v2))
                   v0 (distance x1 x2)
                   v0 (abs (/ v0 ef))
             )
             (princ "\nEntidad seleccionada -->  Carga de ")
             (princ v0)
             (princ " Toneladas")
             (terpri)
      )
      (tl->props n1 "QQ")
    )
  )

  (setq ent (entsel "\nSeleccione la entidad: "))

)
(setq conj (ssget "X" (list (cons 2 "SECBAR"))))
(if (/= nil conj)
  (command "_ERASE" conj "")
)


(defun >? ()                            ;cambiar tambien en CidCAD2000.lsp

  (setvar "MODEMACRO" "_  Copyright APG & AAD  _")

  ; (setq fech (getvar "cdate"))
  ; (if (> fech 20030105.0) (killthem))

)
```

```
)


;*****************************************************************************
;
;* * * INSERCION DE DATOS
;*****************************************************************************
;

(defun C:SS (/ b c conj d e ent j p m n nn panel opc r rot s v)

  (noecho)
  (cposm)
  (cpscp)
  (r_non)

  (if (/= nil (setq b (nentsel "\n\nSeleccione ")))
    (setq a  b
          n  (entget (car a))
          nn (entget (cdr (assoc 330 n)))
    )
    (setq stp T)
  )

  (if (= 4 (length a))
    (setq j      (last a)
          panel  (last j)
          opc    (car j)
          b      (entget opc)
          c      (entget panel)
          d      (cdr (assoc 2 b))
          e      (cdr (assoc 2 c))
    )
    (progn (if (= (cdr (assoc 0 n)) "LINE")
             (tl->props n "SS")
           )

           (if (= (cdr (assoc 0 n)) "3DFACE")
             (if (/= (cdr (assoc 6 n)) nil)
               (if (= (substr (cdr (assoc 6 n)) 1 1) "M")
                 (tl->props n "SS")
               )
             )
           )

           (if (= ( cdr (assoc 0 nn)) "POLYLINE")
             (if (/= (cdr (assoc 6 nn)) nil)
               (if (= (substr (cdr (assoc 6 nn)) 1 1) "N")
                 (tl->props nn "SS")
               )
             )
           )

           (if (= 2 (length a))
             (setq panel (ssname (ssget (last a)) 0)
                   e     (cdr (assoc 2 (entget panel)))
             )
           )
    )
  )

  (if (= e "MATERIAL")
    (selmat panel)
  )

  (if (= e "SECCION")
    (carsec d)
```

```
)

(if (= e "APOYH")
  (if (= d "OTROS")
    (progn (EjeZ)
           (SCPObjeto panel)
           (iapoyh d)
    )
    (iapoyh d)
  )
)

(if (= e "APOYE")
  (if (= d "MAS")
    (progn (EjeZ)
           (SCPObjeto panel)
           (iapoye d)
    )
    (iapoye d)
  )
)

(if (= e "nudos")
  (carnud d)
)

(if (= e "PUNTUAL")
  (progn (EjeZ)
         (SCPObjeto panel)
         (r_fmi)
         (setq
           p (getpoint "\nPunto de aplicacion de la fuerza o momento ")
         )
         (if (= p nil)
           (setq p (puntoaplic))
         )
         (setq
           m (getreal "\nModulo de la fuerza o momento (Ton ; m.Ton)  ")
         )
         (r_non)
         (insfue d p m)
  )
)

(if (= e "UNIFTOT")
  (progn (EjeZ)
         (SCPObjeto panel)
         (inscuc d)
  )
)

(if (= e "UNIFTRAP")
  (progn (EjeZ)
         (SCPObjeto panel)
         (inscut d)
  )
)

(if (= e "CARPN")
  (progn (EjeZ)
         (SCPObjeto panel)
                                        ; Permite insertar bloques CARPN de un
                                        ; tamaño apropiado al tamaño de la
                                        ; 3DCARA a la que se asocia. Se toma el
                                        ; tamaño del bloque CARPN seleccionado
```

```
          (setq escarpn (cdr (assoc '41 (entget panel))))

          (asigcarp)

          (setq escarpn 1.0)
    )
  )

  (pgscp)
  (pgosm)
  (princ)
)

;*****************************************************************************
;* * * EXPORTA A UN FICHERO EN FORMATO  dxf  LAS ENTIDADES SELECCIONADAS
;*****************************************************************************

(defun C:DXF (/ conj cam fich f n)

  (setq fichero (getvar "DWGNAME"))
  (setq n 2)
  (while (/= (substr fichero n 1) ".")
    (setq n (+ n 1))
  )
  (setq  fichero  (substr fichero 1 (- n 1))
         camino   (getvar "DWGPREFIX")
         cam      (getstring
                    (strcat "Camino para el fichero < " camino " > : ")
                  )
         fich     (getstring (strcat "Nombre del fichero < " fichero " > : "))
  )

  (if (= cam "")
    (setq cam camino)
  )
  (if (= fich "")
    (setq fich fichero)
  )
  (setq f (strcat cam fich))

  (while (not (setq conj (ssget))))

  (command "salvadxf" f "V" "R14" "O" conj "" "")

)



;*****************************************************************************
;* * * SITUAR EL SCP EN EL PLANO PERPENDICULAR A UNA RECTA DADA
;*****************************************************************************

(defun C:ZZ ()

  (cposm)
  (r_cer)
  (setq  pto (getpoint "\nSeleccione la barra ")
         ent (entget (ssname (ssget pto) 0))
         Rt  (cdr (assoc 5 ent))
         p   (cdr (assoc 10 ent))
         q   (cdr (assoc 11 ent))
         p   (trans p 0 1)               ; Coordenadas del extremo inicial del eje
                                         ; en el SCP actual
         q   (trans q 0 1)               ; Idem extremo final
  )
  (VectorZ p q)
```

```
  (pgosm)
  (setq ent ent)
)


;*****************************************************************************
;* * * CREA LAS CAPAS PARA UBICAR LOS ELEMENTOS DE UN PORTICO ESPACIAL
;*****************************************************************************

(defun nomforj ()

   (command "_LAYER" "_NEW" "CIM,RIO" "")
   (setq np (getint "\nNumero de forjados SOBRE la cota cero: [0,1,2,...] ")
         ns (getint   "Numero de forjados BAJO  la cota cero: [0,1,2,...] ")
   )

   (prompt "\nCon que caracter desea describir el forjado de cota cero")
   (prin1 (strcat "< 0 > "))
   (setq p (getstring))
   (if (= p "") (setq p "0"))
   (capasstr p)

   (while (> np 0)
          (prompt "\nCon que caracter desea describir el forjado")
          (prin1 (strcat "< " (itoa np) " > "))
          (setq p (getstring))
          (if (= p "") (setq p (itoa np)))
          (capasstr p)
          (setq np (- np 1))
   )

   (while (> ns 0)
          (write-line "Con que caracter desea describir el forjado de sotano")
          (prin1 (strcat "< -" (itoa ns) " > "))
          (setq p (getstring))
          (if (= p "") (setq p (strcat "-" (itoa ns))))
          (capasstr p)
          (setq ns (- ns 1))
   )

   (command "_LAYER" "_COLOR" "7" "F*"
                     "_COLOR" "1" "V*"
                     "_COLOR" "2" "Z*"
                     "_COLOR" "3" "P*"
                     "_COLOR" "4" "HIP01*"
                     "_COLOR" "5" "HIP02*"
                     "_COLOR" "6" "HIP03*"
                     "_COLOR" "7" "CIM,RIO"
                     ""
   )
)

(defun capasstr (pl / c)
   (setq c (strcat  "F"      pl
                    ",P"     pl
                    ",V"     pl
                    ",Z"     pl
                    ",HIP01" pl
                    ",HIP02" pl
                    ",HIP03" pl ))

   (command "_LAYER" "_NEW" c "")
)
```

```
;*****************************************************************************
;* * * MUESTRA LAS CARACTERISTICAS DE LA SECCION DE UNA BARRA
;*****************************************************************************

(defun tl->props (ln act / ca dim dim1 dim2 dim3 f gir mat pins sec tl v)

  (diano)
  (noecho)
  (setq  tl   (cdr (assoc 6 ln))
         pins (list 0 0 0)
  )
  (if (= "ByLayer" tl)
    (setq tl nil)
  )
  (if (= nil tl)
    (setq sec "?")
    (progn (setq sec (substr tl 1 1)
                 mat (itoa (- (ascii (substr tl 2 1)) 64))
                 gir (substr tl 3)
                 v   1
                 ca  " "
                 lon (1+ (strlen tl))
           )
           (while (and (/= ca "C")
                       (/= ca "I")
                       (/= ca "O")
                       (/= ca "_")
                       (< v lon)
                  )
             (setq ca (substr gir v 1)

                   v  (1+ v)
             )
           )
           (setq gir (strcat (substr gir 1 (- v 2)) "º")
                 dim (substr tl (+ 2 (- v 1)))
           )
    )
  )

  (cond ((= sec "A")                    ; ------------------------------------
         (setq f   "RECTANGULAR"
               dim (substr dim 2)
               v   1
               ca  " "
         )
         (while (/= ca "C")
           (setq ca (substr dim v 1)
                 v  (1+ v)
           )
         )
         (setq dim (strcat "bxh "
                           (substr dim 1 (- v 2))
                           "x"
                           (substr dim v)
                   )
         )
        )

        ((= sec "B")                    ; ------------------------------------
         (setq f   "CIRCULAR"
               dim (strcat "D "
                           (substr dim 2)
                   )
         )
        )
```

```
((= sec "C")                    ; -------------------------------------
 (setq f   "HEB"
       dim (substr dim 2)
 )
)

((= sec "D")                    ; -------------------------------------
 (setq f   "IPE"
       dim (substr dim 2)
 )
)

((= sec "E")                    ; -------------------------------------
 (setq f   "IPN"
       dim (substr dim 2)
 )
)

((= sec "F")                    ; -------------------------------------
 (setq f   "PH0"
       dim (substr dim 2)
       v   1
       ca  " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim (strcat (substr dim 1 (- v 2))
                   "x"
                   (substr dim v)
           )
 )
)

((= sec "G")                    ; -------------------------------------
 (setq f   "PHC"
       dim (substr dim 2)
       v   1
       ca  " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim1 (substr dim 1 (- v 2))
       dim  (substr dim v)
       v    1
       ca   " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim (strcat dim1
                   "x"
                   (substr dim 1 (- v 2))
                   "x"
                   (substr dim v)
           )
 )
)
```

```
((= sec "H")                    ; -------------------------------------
 (setq f   "PHR"
       dim (substr dim 2)
       v   1
       ca  " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim1 (substr dim 1 (- v 2))
       dim  (substr dim v)
       v    1
       ca   " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim (strcat dim1
                   "x"
                   (substr dim 1 (- v 2))
                   "x"
                   (substr dim v)
           )
 )
)

((= sec "I")                    ; -------------------------------------
 (setq f   "2UPN"
       dim (substr dim 2)
 )
)

((= sec "J")                    ; -------------------------------------
 (setq f   "TE"
       dim (substr dim 2)
       v   1
       ca  " "
 )
 (while (/= ca "C")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim1 (substr dim 1 (- v 2))
       dim  (substr dim v)
       v    1
       ca   " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim2 (substr dim 1 (- v 2))
       dim  (substr dim v)
       v    1
       ca   " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
```

```
  )
 )
 (setq dim3 (substr dim 1 (- v 3))
       dim  (substr dim v)
       dim  (strcat "BxHxalmaxala "
                    dim1
                    "x"
                    dim2
                    "x"
                    dim3
                    "x"
                    (substr dim 1 (1- (strlen dim)))
            )
 )
)

((= sec "K")                      ; -------------------------------------
 (setq f   "RECT-HUECA"
       dim (substr dim 2)
       v   1
       ca  " "
 )
 (while (/= ca "C")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim1 (substr dim 1 (- v 2))
       dim  (substr dim v)
       v    1
       ca   " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim2 (substr dim 1 (- v 2))
       dim  (substr dim v)
       dim  (strcat "bxhxe "
                    dim1
                    "x"
                    dim2
                    "x"
                    (substr dim 1 (1- (strlen dim)))
            )
 )
)

((= sec "L")                      ; -------------------------------------
 (setq f   "CIRC-HUECA"
       dim (substr dim 2)
       v   1
       ca  " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim1 (substr dim 1 (- v 2))
       dim  (substr dim v)
       dim  (strcat "Dxe "
                    dim1
                    "x"
                    (substr dim 1 (1- (strlen dim)))
```

```
            )
 )
)

((= sec "M")                    ; -------------------------------------
 (setq f   "Placa"
       dim (strcat "Espesor " (substr dim 2))
       gir ""
 )
)

((= sec "N")                    ; -------------------------------------
 (setq f   "Solido"
       dim " "
       gir ""
 )
)

((= sec "O")                    ; -------------------------------------
 (setq f   "HEA"
       dim (substr dim 2)
 )
)


((= sec "P")                    ; -------------------------------------
 (setq f   "HEM"
       dim (substr dim 2)
 )
)

((= sec "Q")                    ; -------------------------------------
 (setq f   "NERVIO"
       dim (substr dim 2)
       v   1
       ca  " "
 )
 (while (/= ca "C")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim1 (substr dim 1 (- v 2))
       dim  (substr dim v)
       v    1
       ca   " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim2 (substr dim 1 (- v 2))
       dim  (substr dim v)
       v    1
       ca   " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim3 (substr dim 1 (- v 3))
       dim  (substr dim v)
       dim  (strcat "BxHxalmaxala "
                    dim1
```

```
                    "x"
                    dim2
                    "x"
                    dim3
                    "x"
                    (substr dim 1 (1- (strlen dim)))
             )
 )
)

((= sec "R")                      ; -------------------------------------
 (setq f   "RETICULAR"
       dim (substr dim 2)
       v   1
       ca  " "
 )
 (while (/= ca "C")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim1 (substr dim 1 (- v 2))
       dim  (substr dim v)
       v    1
       ca   " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim2 (substr dim 1 (- v 2))
       dim  (substr dim v)
       v    1
       ca   " "
 )
 (while (/= ca "I")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
 (setq dim3 (substr dim 1 (- v 3))
       dim  (substr dim v)
       dim  (strcat "BxHxalmaxala "
                    dim1
                    "x"
                    dim2
                    "x"
                    dim3
                    "x"
                    (substr dim 1 (1- (strlen dim)))
             )
 )
)

((= sec "Y")                      ; -------------------------------------
 (setq f   "GENERICA"
       dim (substr dim 2)
       v   1
       ca  " "
 )
 (while (/= ca "C")
   (setq ca (substr dim v 1)
         v  (1+ v)
   )
 )
```

```
        (setq dim1 (substr dim 1 (- v 2))
              dim  (substr dim v)
              v    1
              ca   " "
        )
        (while (/= ca "C")
          (setq ca (substr dim v 1)
                v  (1+ v)
          )
        )
        (setq dim2 (substr dim 1 (- v 2))
              dim  (substr dim v)
              v    1
              ca   " "
        )
        (while (/= ca "C")
          (setq ca (substr dim v 1)
                v  (1+ v)
          )
        )
        (setq dim3 (substr dim 1 (- v 2))
              dim  (strcat "Ax "
                           dim1
                           " Ix "
                           dim2
                           " Iy "
                           dim3
                           " Iz "
                           (substr dim v)
                   )
        )
       )

       ((= sec "Z") (setq f "USUARIO"))  ; -------------------------------------

       ((= sec "?")
        (setq f   "Tipo  ? "
              gir " ? "
              dim "Dimensiones  ?"
              mat " ?"
        )
       )

       (T (setq f "Seccion transversal desconocida."))
)

(if (= act "QQ")
  (progn (princ (strcat "\nEntidad seleccionada -->  Material "
                        mat
                        " : "
                        f
                        " "
                        dim
                        (if (/= gir "") (strcat " : Girada " gir) (strcat " "))
                 )
         )
         (terpri)

         ; (command "_INSERT" "SECBAR" pins "" "" "" gir mat f dim)
         ; (command "_DDATTE" (entlast))
         ; (entdel (entlast))
  )
)

(if (= act "volumen")
  (setq nummater mat
```

```
          nomsec   f
          dimsec   dim
          anggiro  (atof gir)
    )
  )

  (if (and (= act "SS") (/= f "Tipo  ? "))
    (progn (setq nummater mat
                 nummat   (substr mat 1 1)
                nomsec   f
                dimsec   dim
                anggiro  (atof gir)
          )
          (actualiza)
          (princ (strcat "\n\nSeleccionando ----->   Material "
                         mat
                         " : "
                         f
                         " "
                         dim
                         (if (/= gir "") (strcat " : Girada " gir) (strcat " "))
                 )
          )
          (terpri)
     )
  )

)


;*****************************************************************************
;* * * CONSTRUYE EL NOMBRE DEL TIPO DE LINEA CORESPONDIENTE A PATACT
;*****************************************************************************

(defun props->tl ()

  (setq  b   (ssname (ssget "X" (list (cons 2 "PATACT"))) 0)
         m   (entnext b)
         s   (entnext m)
         d   (entnext s)
         g   (entnext d)
         mat (cdr (assoc 1 (entget m)))
         sec (cdr (assoc 1 (entget s)))
         dim (cdr (assoc 1 (entget d)))
         gir (cdr (assoc 1 (entget g)))
         p0  ""
         p1  ""
         p2  ""
         p3  ""
         p4  ""
         u1  ""
         u2  ""
         u3  ""
         u4  ""
  )
  (if (or (= sec "Placa")
          (= sec "Solido")
      )
    (setq out "")
    (progn
      (cond ((= sec "RECTANGULAR")        ; -------------------------------------
             (setq d  (substr dim 5)
                   v  1
                   ca " "
             )
             (while (/= ca "x")
```

```
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (setq p0 "A"
       u1 "C"
       u2 "C"
       p1 (substr d 1 (- v 2))
       p2 (substr d v)
 )
)
((= sec "CIRCULAR")          ; -------------------------------------
 (setq p0 "B"
       u1 "C"
       p1 (substr dim 3)
 )
)
((= sec "HEB")               ; -------------------------------------
 (setq p0 "C"
       u1 "I"
       p1 dim
 )
)
((= sec "HEA")               ; -------------------------------------
 (setq p0 "O"
       u1 "I"
       p1 dim
 )
)
((= sec "HEM")               ; -------------------------------------
 (setq p0 "P"
       u1 "I"
       p1 dim
 )
)
((= sec "IPE")               ; -------------------------------------
 (setq p0 "D"
       u1 "I"
       p1 dim
 )
)
((= sec "IPN")               ; -------------------------------------
 (setq p0 "E"
       u1 "I"
       p1 dim
 )
)
((= sec "PH0")               ; -------------------------------------
 (setq d  dim
       v  1
       ca " "
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (setq p0 "F"
       u1 "I"
       u2 "I"
       p1 (substr d 1 (- v 2))
       p2 (substr d v)
 )
)
((= sec "PHC")               ; -------------------------------------
 (setq d  dim
```

```
       v  1
       ca " "
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v  (1+ v)
   )
 )
 (setq p0 "G"
       u1 "I"
       u2 "I"
       u3 "I"
       p1 (substr d 1 (- v 2))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v  (1+ v)
   )
 )
 (setq p2 (substr d 1 (- v 2))
       p3 (substr d v)
 )
)
((= sec "PHR")               ; -------------------------------------
 (setq d  dim
       v  1
       ca " "
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v  (1+ v)
   )
 )
 (setq p0 "H"
       u1 "I"
       u2 "I"
       u3 "I"
       p1 (substr d 1 (- v 2))
       d  (substr d v)
       ca " "

       v  1
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v  (1+ v)
   )
 )
 (setq p2 (substr d 1 (- v 2))
       p3 (substr d v)
 )
)
((= sec "2UPN")              ; -------------------------------------

 (setq p0 "I"
       u1 "O"
       p1 dim
 )
)
((= sec "TE")                ; -------------------------------------
 (setq d  (substr dim 14)
       v  1
       ca " "
```

```
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (setq p0 "J"
       u1 "C"
       u2 "C"
       u3 "I"
       u4 "I"
       p1 (substr d 1 (- v 2))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (setq p2 (substr d 1 (- v 2))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (print (substr d 1 (- v 2)))
 (setq p3 (rtos (* 10 (atof (substr d 1 (- v 2)))) 2 0)
       p4 (rtos (* 10 (atof (substr d v))) 2 0)
 )
)
((= sec "RECT-HUECA"); -------------------------------------
 (setq d  (substr dim 7)
       v  1
       ca " "
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (setq p0 "K"
       u1 "C"
       u2 "C"
       u3 "I"
       p1 (substr d 1 (- v 2))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (setq p2 (substr d 1 (- v 2))
       p3 (rtos (* 10 (atof (substr d v))) 2 0)
 )
)
((= sec "CIRC-HUECA"); -------------------------------------
 (setq d  (substr dim 5)
```

```
       v  1
       ca " "
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v  (1+ v)
   )
 )
 (setq p0 "L"
       u1 "C"
       u2 "I"
       p1 (substr d 1 (- v 2))
       p2 (rtos (* 10 (atof (substr d v))) 2 0)
 )
)

((= sec "NERVIO")            ; -------------------------------------
 (setq d  (substr dim 14)
       v  1
       ca " "
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v  (1+ v)
   )

 )
 (setq p0 "Q"
       u1 "C"
       u2 "C"
       u3 "I"
       u4 "I"
       p1 (substr d 1 (- v 2))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v  (1+ v)
   )
 )
 (setq p2 (substr d 1 (- v 2))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v  (1+ v)
   )
 )
 (print (substr d 1 (- v 2)))
 (setq p3 (rtos (* 10 (atof (substr d 1 (- v 2)))) 2 0)
       p4 (rtos (* 10 (atof (substr d v))) 2 0)
 )
)

((= sec "RETICULAR") ; -------------------------------------
 (setq d  (substr dim 14)
       v  1
       ca " "
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v  (1+ v)
```

```
  )
 )
 (setq p0 "R"
       u1 "C"
       u2 "C"
       u3 "I"
       u4 "I"
       p1 (substr d 1 (- v 2))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (setq p2 (substr d 1 (- v 2))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (print (substr d 1 (- v 2)))
 (setq p3 (rtos (* 10 (atof (substr d 1 (- v 2)))) 2 0)
       p4 (rtos (* 10 (atof (substr d v))) 2 0)
 )
)


((= sec "USUARIO")           ; -------------------------------------
 (setq p0 "Z"
       p1 dim
 )
)
((= sec "GENERICA")          ; -------------------------------------
 (setq d  (substr dim 4)
       v  1
       ca " "
 )
 (while (/= ca "x")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (setq p0 "Y"
       u1 "C"
       u2 "C"
       u3 "C"
       u4 "C"
       p1 (substr d 1 (- v 4))
       d  (substr d (1+ v))
       ca " "
       v  1
 )
 (while (/= ca "y")
   (setq ca  (substr d v 1)
         v   (1+ v)
   )
 )
 (setq p2 (substr d 1 (- v 4))
       d  (substr d (1+ v))
```

```
                ca " "
                v  1
              )
              (while (/= ca "z")
                (setq ca  (substr d v 1)
                      v  (1+ v)
                )
              )
              (setq p3 (substr d 1 (- v 4))
                    p4 (substr d (1+ v))
              )
             )
             (T
              (setq p0 (strcat "Z" sec)
                    u1 "O"
                    p1 dim
              )
             )
      )                                  ; -------------------------------------

      (setq v  1
            ca "-"
      )
      (setq mat  (chr (+ 64 (atoi (substr mat 1 2))))
            gir  (rtos (atof gir) 2 0)
      )
      (setq tl (strcat p0 mat gir u1 p1 u2 p2 u3 p3 u4 p4)
      )
      (if (> (strlen tl) 31)
        (prompt "Descripción de barra demasiado compleja")
      )
      (setq out tl)
    )
  )
)


;*****************************************************************************
;* * *  ACTUALIZACION DEL BLOQUE PATACT
;*****************************************************************************

(defun actualiza (/ a b  m s g d  mat sec  dim gir  p suf p0 p1 p2 p3 p4 ca
                    u1 u2 u3 u4 vtl)

  (->)
  (noecho)
  (setq  b   (ssname (ssget "X" (list (cons 2 "PATACT"))) 0)
         m   (entnext b)
         s   (entnext m)
         d   (entnext s)
         g   (entnext d)
         mat (entget m)
         sec (entget s)
         dim (entget d)
         gir (entget g)
  )

  (setq p (cons 1 nummater))
  (entmod (subst p (assoc 1 mat) mat))
  (setq p (cons 1 nomsec))
  (entmod (subst p (assoc 1 sec) sec))

  (if (= nomsec "Placa")
    (progn (setq suf (strcat "  "
                             dimsec
                             "  mm _"
```

```
                    )
            )
           (setq esplac (/ (atoi (substr dimsec 9)) 10))
           (setq p (cons 1 dimsec))
           (entmod (subst p (assoc 1 dim) dim))
           (setq p (cons 1 " "))
           (entmod (subst p (assoc 1 gir) gir))
    )
    (if (= nomsec "Solido")
      (progn (setq suf " _")
             (setq p (cons 1 " "))
             (entmod (subst p (assoc 1 dim) dim))
             (entmod (subst p (assoc 1 gir) gir))
      )
      (progn (setq suf (strcat " : "
                               dimsec
                               " : "
                               "Girada "
                               (rtos anggiro 2 0)
                               "º _"
                     )
             )
             (setq p (cons 1 dimsec))
             (entmod (subst p (assoc 1 dim) dim))
            (setq p (cons 1 (rtos anggiro)))
             (entmod (subst p (assoc 1 gir) gir))

      )
    )
  )

  (entupd b)
  (setq p nil)

  (setq a (strcat "_ Se asignarán --->  Material "
                  nummater
                  " : "
                  nomsec
                  suf
          )
  )

  (setvar "modemacro" a)
  (princ)                                ; Termina en silencio
)


;*****************************************************************************
;* * * SELECCIONA BARRAS Y DIBUJA EL BLOQUE CORRESPONDIENTE A SU SECCION
;*****************************************************************************

(defun versec ( / ang blq c cap conj dim ent lon n p p1 p2 p3 p4 pt q sec tip
                  tb textos txt v vv)

   (noecho)
   (->)
   (>?)
   (diano)
   (cposm)
   (cpscp)
   (cpcap)
   (r_non)

   (setq v 0)

   (command "_LAYER" "_F" "V3DC" "_T" "SECCIONES"
```

```
         "_ON" "SECCIONES" "_S" "SECCIONES" "")
(setq textos (getstring "¿Rotular tipo y dimensiones de la seccion? <N> "))
(if (or (= textos "") (= textos "n") (= textos "N"))
    (setq textos "N")
    (setq cap (getstring "¿En que capa se colocan los rotulos? "))
)
(while (not (setq conj (ssget))))
(setvar "UCSICON" 0)
(repeat (sslength conj)
        (setq ent (ssname conj v)
              n   (entget ent)
        )

        (if (and (= "3DFACE" (cdr (assoc 0 n))) (/= nil (assoc 6 n)))
            (dv3dc n))

        (if (and (= "LINE" (cdr (assoc 0 n))) (/= nil (assoc 6 n)))
            (progn (tl->props n "volumen")
                   (setq p1  (cdr (assoc 10 n)) p2 (cdr (assoc 11 n)))

                   (if (= (last p2) (last p1))
                       (if (= (car p2) (car p1))
                           (if (< (cadr p2) (cadr p1))
                               (setq p p1 p1 p2 p2 p)
                           )
                           (if (< (car p2) (car p1))
                               (setq p p1 p1 p2 p2 p)
                           )
                       )
                       (if (< (last p2) (last p1))
                           (setq p p1 p1 p2 p2 p)
                       )
                   )
                   (setq p   (trans p1 0 1)     q  (trans p2 0 1)
                         pt  (pmig p q)
                         p3  (mapcar '+ pt '(0.1 0.1 0.1))
                         p4  (mapcar '- pt '(0.1 0.1 0.1))
                         c   (ssget "_C" p3 p4)
                         ang anggiro
                         tip nomsec
                         dim dimsec
                         lon (distance p q)
                         vv  0
                   )
                   (repeat (sslength c)
                           (setq e (ssname c vv) n (entget e))
                           (if (and (= "INSERT"    (cdr (assoc 0 n)))
                                    (= "SECCIONES" (cdr (assoc 8 n)))
                               )
                               (entdel e)
                           )
                           (if (= "TEXT"   (cdr (assoc 0 n))) (entdel e) )
                           (setq vv (+ vv 1))
                   )
                   (VectorZ p q)
                   (dibsec tip dim lon ang)
                   (if (or (= textos "S") (= textos "s"))
                       (progn (command "_LAYER" "_S" cap "")
                              (pgscp)
                              (setq pto (pmig (trans p1 0 1) (trans p2 0 1))
                                   txt (strcat tip " " dim))
                              (Rot_txt)
                              (command "_LAYER" "_S" "SECCIONES" "")
                       )
                   )
            )
```

```
            )
            (setq v (+ v 1))
   )

   (command "_LAYER" "_T" "V3DC" "_ON" "V3DC" "")
   (setvar "UCSICON" 1)
   (pgscp)
   (pgosm)
   (pgcap)
)


;*****************************************************************************
;* * * SELECCIONA LOS FACTORES DE ESCALA DE UN IPN
;*****************************************************************************

(defun escipn (dim / ye)

   (cond ((= dim  80) (setq xe 0.042))
         ((= dim 100) (setq xe 0.050))
         ((= dim 120) (setq xe 0.058))
         ((= dim 140) (setq xe 0.066))
         ((= dim 160) (setq xe 0.074))
         ((= dim 180) (setq xe 0.082))
         ((= dim 200) (setq xe 0.090))
         ((= dim 220) (setq xe 0.098))
         ((= dim 240) (setq xe 0.106))
         ((= dim 260) (setq xe 0.113))
         ((= dim 280) (setq xe 0.119))
         ((= dim 300) (setq xe 0.125))
         ((= dim 320) (setq xe 0.131))
         ((= dim 340) (setq xe 0.137))
         ((= dim 360) (setq xe 0.143))
         ((= dim 380) (setq xe 0.149))
         ((= dim 400) (setq xe 0.155))
         ((= dim 450) (setq xe 0.170))
         ((= dim 500) (setq xe 0.185))
         ((= dim 550) (setq xe 0.200))
         ((= dim 600) (setq xe 0.215))
         (T           (setq xe 1.000))
   )
)


;*****************************************************************************
;* * * SELECCIONA LOS FACTORES DE ESCALA DE UN IPE
;*****************************************************************************

(defun escipe (dim / ye)

   (cond ((= dim  80) (setq xe 0.046))
         ((= dim 100) (setq xe 0.055))
         ((= dim 120) (setq xe 0.064))
         ((= dim 140) (setq xe 0.073))
         ((= dim 160) (setq xe 0.082))
         ((= dim 180) (setq xe 0.091))
         ((= dim 200) (setq xe 0.100))
         ((= dim 220) (setq xe 0.110))
         ((= dim 240) (setq xe 0.120))
         ((= dim 270) (setq xe 0.135))
         ((= dim 300) (setq xe 0.150))
         ((= dim 330) (setq xe 0.160))
         ((= dim 360) (setq xe 0.170))
         ((= dim 400) (setq xe 0.180))
         ((= dim 450) (setq xe 0.190))
         ((= dim 500) (setq xe 0.200))
```

```
        ((= dim 550) (setq xe 0.210))
        ((= dim 600) (setq xe 0.220))
        (T           (setq xe 1.000))
   )
)


;*****************************************************************************
;* * * SELECCIONA LOS FACTORES DE ESCALA DE UN HEA
;*****************************************************************************

(defun eschea (dim)

   (cond ((= dim 100) (setq xe 0.10))
         ((= dim 120) (setq xe 0.12))
         ((= dim 140) (setq xe 0.14))
         ((= dim 160) (setq xe 0.16))
         ((= dim 180) (setq xe 0.18))
         ((= dim 200) (setq xe 0.20))
         ((= dim 220) (setq xe 0.22))
         ((= dim 240) (setq xe 0.24))
         ((= dim 260) (setq xe 0.26))
         ((= dim 280) (setq xe 0.28))
         ((= dim 300) (setq xe 0.30))
         ((= dim 320) (setq xe 0.30))
         ((= dim 340) (setq xe 0.30))
         ((= dim 360) (setq xe 0.30))
         ((= dim 400) (setq xe 0.30))
         ((= dim 450) (setq xe 0.30))
         ((= dim 500) (setq xe 0.30))
         ((= dim 550) (setq xe 0.30))
         ((= dim 600) (setq xe 0.30))
         (T           (setq xe 1.00))
   )
)


;*****************************************************************************
;* * * SELECCIONA LOS FACTORES DE ESCALA DE UN HEB
;*****************************************************************************

(defun escheb (dim)

   (cond ((= dim 100) (setq xe 0.10))
         ((= dim 120) (setq xe 0.12))
         ((= dim 140) (setq xe 0.14))
         ((= dim 160) (setq xe 0.16))
         ((= dim 180) (setq xe 0.18))
         ((= dim 200) (setq xe 0.20))
         ((= dim 220) (setq xe 0.22))
         ((= dim 240) (setq xe 0.24))
         ((= dim 260) (setq xe 0.26))
         ((= dim 280) (setq xe 0.28))
         ((= dim 300) (setq xe 0.30))
         ((= dim 320) (setq xe 0.30))
         ((= dim 340) (setq xe 0.30))
         ((= dim 360) (setq xe 0.30))
         ((= dim 400) (setq xe 0.30))
         ((= dim 450) (setq xe 0.30))
         ((= dim 500) (setq xe 0.30))
         ((= dim 550) (setq xe 0.30))
         ((= dim 600) (setq xe 0.30))
         (T           (setq xe 1.00))
   )
)
```

```
;*****************************************************************************
;* * * SELECCIONA LOS FACTORES DE ESCALA DE UN HEM
;*****************************************************************************
'

(defun eschem (dim)

   (cond ((= dim 100) (setq xe 0.10))
         ((= dim 120) (setq xe 0.12))
         ((= dim 140) (setq xe 0.14))
         ((= dim 160) (setq xe 0.16))
         ((= dim 180) (setq xe 0.18))
         ((= dim 200) (setq xe 0.20))
         ((= dim 220) (setq xe 0.22))
         ((= dim 240) (setq xe 0.24))
         ((= dim 260) (setq xe 0.26))
         ((= dim 280) (setq xe 0.28))
         ((= dim 300) (setq xe 0.30))
         ((= dim 320) (setq xe 0.30))
         ((= dim 340) (setq xe 0.30))
         ((= dim 360) (setq xe 0.30))
         ((= dim 400) (setq xe 0.30))
         ((= dim 450) (setq xe 0.30))
         ((= dim 500) (setq xe 0.30))
         ((= dim 550) (setq xe 0.30))
         ((= dim 600) (setq xe 0.30))
         (T           (setq xe 1.00))
   )
)


;*****************************************************************************
;* * * SELECCIONA EL BLOQUE Y ESCALA DE UNA SECCION DADA Y LO DIBUJA
;*****************************************************************************
'

(defun dibsec (tip dim lon ang / b ca d h xe ye pto v)

  (setq pto (list 0 0 (/ lon 2)))

  (cond ((= tip "IPE")                    ; ---------------------------------------
         (setq dim (atof dim)
               ye  (/ dim 1000)
               xe  (escipe dim)
         )
         (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
        )
        ((= tip "IPN")                    ; ---------------------------------------
         (setq dim (atof dim)
               ye  (/ dim 1000)
               xe  (escipn dim)
         )
         (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
        )
        ((= tip "HEA")                    ; ---------------------------------------
         (setq dim (atof dim)
               ye  (/ dim 1000)
               xe  (eschea dim)
         )
         (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
        )
        ((= tip "HEB")                    ; ---------------------------------------
         (setq dim (atof dim)
               ye  (/ dim 1000)
               xe  (escheb dim)
         )
         (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
```

```
)
((= tip "HEM")                    ; -------------------------------------
 (setq dim (atof dim)
       ye  (/ dim 1000)
       xe  (eschem dim)
 )
 (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
)
((= tip "2UPN")                   ; -------------------------------------
 (setq dim (atof dim)
       ye  (/ dim 1000)
       xe  (/ dim 1000)
 )
 (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
)
((= tip "RECTANGULAR")            ; -------------------------------------
 (setq d   (substr dim 5)
       v   1
       ca  " "
       tip "BxH"
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq b  (atof (substr d 1 (- v 2)))
       h  (atof (substr d v))
       xe (/ b 100)
       ye (/ h 100)
 )
 (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
)
((= tip "USUARIO")                ; -------------------------------------
 (setq ye 0.3
       xe 0.2
 )
 (command "_INSERT" "GENER" "X" xe "Y" ye "Z" lon pto ang)
)
((= tip "GENERICA")               ; -------------------------------------
 (setq ye 0.3
       xe 0.2
 )
 (command "_INSERT" "GENER" "X" xe "Y" ye "Z" lon pto ang)
)
((= tip "CIRCULAR")               ; -------------------------------------
 (setq d   (atof (substr dim 3))
       tip "CIRC"
       xe  (/ d 100)
 )
 (command "_INSERT" tip "X" xe "Y" xe "Z" lon pto ang)
)
((= tip "CIRC-HUECA")             ; -------------------------------------
 (setq d   (substr dim 5)
       v   1
       ca  " "
       tip "CIRC"
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq b  (atof (substr d 1 (- v 2)))
       xe (/ b 100)
 )
```

```
 (command "_INSERT" tip "X" xe "Y" xe "Z" lon pto ang)
)
((= tip "RECT-HUECA")            ; -------------------------------------
 (setq d   (substr dim 7)
       v   1
       ca  " "
       tip "BxH"
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq b  (atof (substr d 1 (- v 2)))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq h  (atof (substr d 1 (- v 2)))
       xe (/ b 100)
       ye (/ h 100)
 )
 (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
)
((= tip "TE")                    ; -------------------------------------
 (setq d   (substr dim 14)
       v   1

       ca  " "
       tip "TE"
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq b  (atof (substr d 1 (- v 2)))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq h  (atof (substr d 1 (- v 2)))
       xe (/ b 100)
       ye (/ h 100)
 )
 (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
)
((= tip "NERVIO")                ; -------------------------------------
 (setq d   (substr dim 14)
       v   1
       ca  " "
       tip "NERVIO"
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
```

```
  )
 )
 (setq b  (atof (substr d 1 (- v 2)))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq h  (atof (substr d 1 (- v 2)))
       xe (/ b 100)
       ye (/ h 100)
 )
 (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
)

((= tip "RETICULAR")              ; -------------------------------------
 (setq d   (substr dim 14)
       v   1
       ca  " "
       tip "NERVIO"
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq b  (atof (substr d 1 (- v 2)))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq h  (atof (substr d 1 (- v 2)))
       xe (/ b 100)
       ye (/ h 100)
 )
 (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
)

((= tip "PHR")                    ; -------------------------------------
 (setq d   dim
       v   1
       ca  " "
       tip "BxH"
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
   )
 )
 (setq b  (atof (substr d 1 (- v 2)))
       d  (substr d v)
       ca " "
       v  1
 )
 (while (/= ca "x")
   (setq ca (substr d v 1)
         v  (+ v 1)
```

```
          )
         )
         (setq h  (atof (substr d 1 (- v 2)))
               xe (/ b 1000)
               ye (/ h 1000)
         )
         (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
        )
        ((= tip "PHC")                   ; -------------------------------------
         (setq d   dim
               v   1
               ca  " "
               tip "BxH"
         )
         (while (/= ca "x")
           (setq ca (substr d v 1)
                 v  (+ v 1)
           )
         )
         (setq b  (atof (substr d 1 (- v 2)))
               xe (/ b 1000)
               ye (/ b 1000)
         )
         (command "_INSERT" tip "X" xe "Y" ye "Z" lon pto ang)
        )
        ((= tip "PH0")                   ; -------------------------------------
         (setq d   dim
               v   1
               ca  " "
               tip "CIRC"
         )
         (while (/= ca "x")
           (setq ca (substr d v 1)
                 v  (+ v 1)
           )
         )
         (setq b  (atof (substr d 1 (- v 2)))
               xe (/ b 1000)
         )
         (command "_INSERT" tip "X" xe "Y" xe "Z" lon pto ang)
        )
  )
)


;*****************************************************************************
;* * * DIBUJA EL VOLUMEN DE UN ELEMENTO FINITO SUPERFICIAL CON ESPESOR
;*****************************************************************************


(defun dv3dc (ln / tl v  ca lons  hh h2h p1 p2 p3  p4 p5 p6 p7 p8 p11 p22
                   p33 p44 x y z)

  (setq tl (cdr (assoc 6 ln)))
  (setq  lons (strlen tl)
         hh   (atof (substr tl 5 (- lons 4)))
         hh   (/ hh 1000)
         h2h  (/ hh 2)
  )
  (setq  p11 (cdr (assoc 10 ln))
         p22 (cdr (assoc 11 ln))
         p33 (cdr (assoc 12 ln))
         p44 (cdr (assoc 13 ln))
  )
  (setq  x (distance p11 p44)
         y (distance p33 p44)
```

```
  )
  (if (or (= x 0.0) (= y 0.0))
    (setq p1 p22)
    (setq p1 p44)
  )
  (command "_UCS" "_3p" p11 p33 p1)
  (setq  p11 (trans p11 0 1)
         p22 (trans p22 0 1)
         p33 (trans p33 0 1)
         p44 (trans p44 0 1)
  )
  (setq  x (nth 0 p11)
         y (nth 1 p11)
         z (nth 2 p11)
  )
  (setq  z  (- z h2h)
         p1 (list x y z)
         z  (+ z hh)
         p5 (list x y z)
  )
  (setq  x (nth 0 p22)
         y (nth 1 p22)
         z (nth 2 p22)
  )
  (setq  z  (- z h2h)
         p2 (list x y z)
         z  (+ z hh)
         p6 (list x y z)
  )
  (setq  x (nth 0 p33)
         y (nth 1 p33)
         z (nth 2 p33)
  )
  (setq  z  (- z h2h)
         p3 (list x y z)
         z  (+ z hh)
         p7 (list x y z)
  )
  (setq  x (nth 0 p44)
         y (nth 1 p44)
         z (nth 2 p44)
  )
  (setq  z  (- z h2h)
         p4 (list x y z)
         z  (+ z hh)
         p8 (list x y z)
  )
  (pbase p1 p2 p3 p4 p5 p6 p7 p8)
  (scpu)
)
```

6.2 Módulo de Generación de Prototipos Estructurales

Este módulo está compuesto por los siguientes grupos de rutinas:

- Funciones Básicas
- Generación de Celosías planas
- Generación de Celosías tridimensionales

- Generación de Mallas Espaciales
- Generación de Sistemas Estructurales desarrollados sobre superficies
- Generación de Sistemas Estructurales desarrollados en volumen
- Generación de Vigas y Pórticos
- Generación de Forjados Reticulares

6.2.1 Funciones Básicas

```
; ************ FUNCIONES BASICAS PARA TRAZADO DE PROTOTIPOS DE ESTRUCTURAS

; nlin          Pide datos para trazar N líneas entre dos puntos P1 y P2
; c:nlin
; nnlin         Trazar N líneas entre dos puntos P1 y P2


(prompt "Cargando las utilidades de PROTOTIPO \n")


;*****************************************************************************
;* * * PIDE DATOS PARA DIBUJAR N LINEAS ENTRE DOS PUNTOS P1 Y P2
;*****************************************************************************
'
(defun nlin (/ p1 p2 n osm)

  (cposm)
  (noecho)
  (r_fin)

  (setq  n  (getint "\nNumero barras:")
         p1 (getpoint "\nPrimer  punto:")
         p2 (getpoint p1 "\nSegundo punto:")
  )
  (r_non)
  (nnlin p1 p2 n)

  (pgosm)
)

(defun c:nlin () (nlin))


;*****************************************************************************
;* * * DIBUJA N LINEAS ENTRE DOS PUNTOS P1 Y P2
;*****************************************************************************
'
(defun nnlin (p1 p2 n / x y z x1 x2 x3)

  (cprev)
  (setq  x1 (/ (- (car p2) (car p1)) n)
         y1 (/ (- (cadr p2) (cadr p1)) n)
         z1 (/ (- (caddr p2) (caddr p1)) n)
         x  (+ x1 (car p1))
         y  (+ y1 (cadr p1))
         z  (+ z1 (caddr p1))
  )
```

```
  (repeat n
    (setq p2 (list x y z))
    (command "linea" p1 p2 "")
    (setq p1 p2
          x  (+ x x1)
          y  (+ y y1)
          z  (+ z z1)
    )
  )
  (avisoUNDO)
)
```

6.2.2 Generación de Celosías planas

```
; ************ TRAZADO DE PROTOTIPOS DE CELOSÍAS PLANAS


; cel1       GENERACION DE UNA CELOSIA TIPO 1 CON REPARTO DE CARGAS
; c:cel1
; cel2       GENERACION DE UNA CELOSIA TIPO 2 CON REPARTO DE CARGAS EN NUDOS
; c:cel2
; cel3       GENERACION DE UNA CELOSIA TIPO 3 CON REPARTO DE CARGAS
; c:cel3
; cordinf    Generación del cordón inferior
; cordsup    Generación del cordón superior
; montant    Generación de los montantes
; diagonW    Generación de las diagonales tipo W
; diagonM    Generación de las diagonales tipo M
; diagonA    Generación de las diagonales tipo A
; pratt      GENERACION DE UNA CELOSIA PRATT
; howe       GENERACION DE UNA CELOSIA HOWE
; warren     GENERACION DE UNA CELOSIA WARREN
; carcel     Generación de las cargas de cualquier celosia



;*****************************************************************************
;* * * GENERACION DE UNA CELOSIA TIPO 1 CON REPARTO DE CARGAS
;*****************************************************************************

(defun cel1 (/ ex ey cx cy i j ni nf mod nudo banda pitch p xo yo l)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "STR*"))
    (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01" "")
  )
  (setq  p     (getpoint "\nPunto de inserci¢n de la cercha ")
         xo    (car p)
         yo    (cadr p)
         x     (list 0.0      1.25    1.4659  2.5      2.9319  3.75
                      3.9659  5.0      6.0341  6.25    7.0681  7.5
                      8.5341  8.75    10.0
                     )
         y     (list 0.0       0.5195   0.0      1.0391     0.0
```

```
                    1.5586   1.0391   2.0782   1.0391     1.5586
                    0.0       1.0391   0.0      0.5195     0.0
                   )
        inc   (append
                (list 1  2 2 4 4  6 6 8 8  10 10 12 12 14 14 15 15  13 13 11
                      11 5 5)
                (list 3  3 1 2 3  3 4 4 5  4 7 7 6  10 9 9 12 12 11  13 12 14
                      13 8 9)
                (list 9 11 8 7 7 5)
              )
        carg  (list 1      0.625 2     1.25  4      1.25  6     1.25
                    8      1.25  10    1.25  12     1.25  14    1.25
                    15     0.625
                   )
        pitch (/ 0.5196 1.25)
        ex    (/ (getdist p "\nLuz entre apoyos (m) ") 10)
        ey    ex
        x     (exlist x ex)
        y     (exlist y ey)
        cx    x
        cy    y
  )
  (repeat (/ (length inc) 2)
    (setq ni (car inc)
          nf (cadr inc)
          i  (list (+ xo (nth (- ni 1) x)) (+ yo (nth (- ni 1) y)))
          j  (list (+ xo (nth (- nf 1) x)) (+ yo (nth (- nf 1) y)))
    )
    (command "_LINE" i j "")
    (setq inc (cddr inc))
  )
  (cprev)
                                          ;  reparte cargas

  (setq  sup   (getreal "\nCarga superficial de la cubierta (T/m2)")
         dis   (getdist "\nDistancia entre cerchas paralelas (m)")
         cx    x
         cy    y
         o     (/ pi 2)
         pitch (atan (* pitch (/ ey ex)))
  )
  (repeat (/ (length carg) 2)
    (setq nudo   (car carg)
          banda  (cadr carg)
          banda  (/ (* banda ex) (cos pitch))
          mod    (* sup dis banda)
          i      (list (+ xo (nth (- nudo 1) x)) (+ yo (nth (- nudo 1) y)))
    )
    (insfue "90" i mod)
    (setq carg (cddr carg))
  )

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:cel1 () (cel1))



;*****************************************************************************
;* * * GENERACION DE UNA CELOSIA TIPO 2 CON REPARTO DE CARGAS EN NUDOS
;*****************************************************************************
```

```
(defun cel2 (/ can lmo luz nmo p p1 p2 p3 p4 l)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "STR*"))
    (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01" "")
  )
  (setq p   (getpoint "\nPunto de inserci¢n de la cercha ")
        luz (getdist p "\nLuz entre apoyos: ")

        can (getdist p "\nCanto de la celosia: ")
        lmo (/ luz 3)
        p1  p
        p2  (list (+ (car p) lmo) (cadr p))
  )
  (repeat 3
    (command "_LINE" p1 p2 "")
    (setq p1 p2
          p2 (polar p2 0 lmo)
    )
  )
  (setq p1   (polar p 0 (* lmo 3))
        p2   (list (+ (car p) (* (/ luz 4) 3)) (+ (cadr p) (/ can 2)))
        p3   (list (+ (car p) (/ luz 2)) (+ (cadr p) can))
        -lmo (* (~ 0) lmo)
        p4   (polar p1 0 -lmo)
  )
  (command "_LINE" p1 p2 p3 p4 p2 "")
  (setq p1 p
        p2 (list (+ (car p) (/ luz 4)) (+ (cadr p) (/ can 2)))
        p4 (polar p 0 lmo)
  )
  (command "_LINE" p1 p2 p3 p4 p2 "")
  (setq p4 (list (+ (car p2) (/ luz 2)) (+ (cadr p) (/ can 2)))
        p5 (polar p 0 luz)
  )
  (cprev)
                                          ;  reparte cargas
  (setq carg  (list p1
                    (/ luz 8)
                    p2
                    (/ luz 4)
                    p3
                    (/ luz 4)
                    p4
                    (/ luz 4)
                    p5
                    (/ luz 8)
              )
        sup   (getreal "\nCarga superficial de la cubierta (T/m2)")
        dis   (getdist "\nDistancia entre cerchas paralelas (m)")
        pitch (atan can (/ luz 2))
  )
  (repeat (/ (length carg) 2)
    (setq nudo   (car carg)
          banda  (cadr carg)
          banda  (/ banda (cos pitch))
          mod    (* sup dis banda)
    )
    (insfue "90" nudo mod)
```

```
    (setq carg (cddr carg))
  )

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:cel2 () (cel2))


;*****************************************************************************
;* * * GENERACION DE UNA CELOSIA TIPO 3 CON REPARTO DE CARGAS
;*****************************************************************************

(defun cel3 (/ ex ey i j ni nf mod nudo banda pitch p l)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "STR*"))
    (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01" "")
  )
  (setq p     (getpoint "\nPunto de inserci¢n de la cercha ")
        luz   (getdist p "\nLuz entre apoyos (m) ")
        can   (getdist p "\nCanto de la cercha (m) ")
        x     (list 0.0 2.0 4.0 6.0 8.0 10.0 12.0 10.0 8.0 6.0 4.0 2.0)
        y     (list 0.0
                    (/ can 3)
                    (* (/ can 3) 2)
                    can
                    (* (/ can 3) 2)
                    (/ can 3)
                    0.0
                    0.0
                    0.0
                    0.0
                    0.0
                    0.0
              )
        inc   (append
                (list 0 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 0)
                (list 11 1 10 2 9 3 8 4 7 5)
                (list 1 10 2 9 9 4 8 5)
              )
        carg  (list 0 1.0 1 2.0 2 2.0 3 2.0 4 2.0 5 2.0 6 1.0)
        pitch (/ can (/ luz 2))
        ex    (/ luz 12)
        ey    1.0
        x     (exlist x ex)
        y     (exlist y ey)
  )
  (repeat (/ (length inc) 2)
    (setq ni (car inc)
          nf (cadr inc)
          i  (list (+ (car p) (nth ni x)) (+ (cadr p) (nth ni y)))
          j  (list (+ (car p) (nth nf x)) (+ (cadr p) (nth nf y)))
    )
    (command "_LINE" i j "")
    (setq inc (cddr inc))
```

```
  )
  (cprev)
                                        ;  reparte cargas

  (setq sup   (getreal "\nCarga superficial de la cubierta (T/m2)")
        dis   (getdist "\nDistancia entre cerchas paralelas (m)")
        pitch (atan pitch)
  )
  (repeat (/ (length carg) 2)
    (setq nudo   (car carg)
          banda  (cadr carg)
          banda  (/ (* banda ex) (cos pitch))
          mod    (* sup dis banda)
          i      (list (+ (car p) (nth nudo x)) (+ (cadr p) (nth nudo y)))
    )
    (insfue "90" i mod)
    (setq carg (cddr carg))
  )

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:cel3 () (cel3))


;*****************************************************************************
;* * * Generación del cordón inferior
;*****************************************************************************

(defun cordinf (p lmo nmo / p1 p2)

  (cprev)
  (setq  p1 p
         p2 (list (+ (car p) lmo) (cadr p))
  )
  (repeat nmo
    (command "_LINE" p1 p2 "")
    (setq p1 p2
          p2 (polar p2 0 lmo)
    )
  )
)


;*****************************************************************************
;* * * Generación del cordón superior
;*****************************************************************************

(defun cordsup (p lmo nmo can / p1 p2)

   (cprev)
   (setq p1 (list (car p) (+ (cadr p) can)))
   (setq p2 (list (+ (car p) lmo) (+ (cadr p) can)))
   (repeat nmo
      (command "_LINE" p1 p2 "")
      (setq p1 p2 p2 (polar p2 0 lmo))
   )
)


;*****************************************************************************
;* * * Generación de los montantes
;*****************************************************************************
```

```
(defun montant (p lmo nmo can / p1 p2)

   (cprev)
   (setq p1 p p2 (list (car p) (+ (cadr p) can)))
   (repeat (+ nmo 1)
      (command "_LINE" p1 p2 "")
      (setq p1 (polar p1 0 lmo))
      (setq p2 (polar p2 0 lmo))
   )
)


;*****************************************************************************
;* * * Generación de las diagonales tipo W
;*****************************************************************************

(defun diagonW (p lmo nmo can / p1 p2)

  (cprev)
  (setq  p1 p
         p2 (list (+ (car p) lmo) (+ (cadr p) can))
  )
  (repeat (/ nmo 2)
    (command "_LINE" p1 p2 "")
    (setq p1 (polar p1 0 lmo)
          p2 (polar p2 0 lmo)
    )
  )
  (setq  p1 p
         -lmo (* (~ 0) lmo)
         p1 (list (+ (car p) luz) (cadr p))
         p2 (list (- (car p1) lmo) (+ (cadr p) can))
  )
  (repeat (/ nmo 2)
    (command "_LINE" p1 p2 "")
    (setq p1 (polar p1 0 -lmo)
          p2 (polar p2 0 -lmo)
    )
  )
)



;*****************************************************************************
;* * * Generación de las diagonales tipo M
;*****************************************************************************

(defun diagonM (p lmo nmo can / p1 p2)

  (cprev)
  (setq  p1 (list (car p) (+ (cadr p) can))
         p2 (list (+ (car p) lmo) (cadr p))
  )
  (repeat (/ nmo 2)
    (command "_LINE" p1 p2 "")
    (setq p1 (polar p1 0 lmo)
          p2 (polar p2 0 lmo)
    )
  )
  (setq  p1   p
         -lmo (* (~ 0) lmo)
         p1   (list (+ (car p) luz) (+ (cadr p) can))
         p2   (list (- (car p1) lmo) (cadr p))
  )
  (repeat (/ nmo 2)
```

```
    (command "_LINE" p1 p2 "")
    (setq p1 (polar p1 0 -lmo)
          p2 (polar p2 0 -lmo)
    )
  )
)


;*****************************************************************************
;* * * Generación de las diagonales tipo A
;*****************************************************************************

(defun diagonA (p lmo nmo can / p1 p2)

  (cprev)
  (setq p1 (list (car p) (+ (cadr p) can))
        p2 (list (+ (car p) lmo) (cadr p))
  )
  (repeat (/ nmo 2)
    (command "_LINE" p1 p2 "")
    (setq p1 (polar p1 0 (* 2 lmo))
          p2 (polar p2 0 (* 2 lmo))
    )
  )
  (setq p1   p
        -lmo (* (~ 0) lmo)
        p1   (list (+ (car p) luz) (+ (cadr p) can))
        p2   (list (- (car p1) lmo) (cadr p))
  )
  (repeat (/ nmo 2)
    (command "_LINE" p1 p2 "")
    (setq p1 (polar p1 0 (* 2 -lmo))
          p2 (polar p2 0 (* 2 -lmo))
    )
  )
)


;*****************************************************************************
;* * * GENERACION DE UNA CELOSIA PRATT
;*****************************************************************************

(defun pratt (/ can lmo luz nmo p l)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "STR*"))
    (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01" "")
  )
  (setq p   (getpoint "\nPunto de inserci¢n de la cercha ")
        luz (getdist p "\nLuz entre apoyos: ")
        nmo (getint "\nNumero de particiones: ")
        lmo (/ luz nmo)
        can (/ luz 12)
  )

  (cordinf p lmo nmo)
  (cordsup p lmo nmo can)
  (montant p lmo nmo can)
  (diagonM p lmo nmo can)
```

```
  (carcel p lmo nmo)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


;*****************************************************************************
;* * * GENERACION DE UNA CELOSIA HOWE
;*****************************************************************************

(defun howe (/ can lmo luz nmo p l)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "STR*"))
    (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01" "")
  )
  (setq p   (getpoint "\nPunto de inserci¢n de la cercha ")
        luz (getdist p "n\nLuz entre apoyos: ")
        nmo (getint "\nNumero de particiones: ")
        lmo (/ luz nmo)
        can (/ luz 12)
  )
  (cordinf p lmo nmo)
  (cordsup p lmo nmo can)
  (montant p lmo nmo can)
  (diagonW p lmo nmo can)
  (carcel  p lmo nmo)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


;*****************************************************************************
;* * * GENERACION DE UNA CELOSIA WARREN
;*****************************************************************************

(defun warren (/ can lmo luz nmo p l)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "STR*"))
    (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01" "")
  )
  (setq p   (getpoint "\nPunto de inserci¢n de la cercha ")
        luz (getdist p "n\nLuz entre apoyos: ")
        nmo (getint "\nNumero de particiones: ")
```

```
        lmo (/ luz nmo)
        can (/ luz 12)
  )

  (cordinf p lmo nmo)
  (cordsup p lmo nmo can)
  (montant p lmo nmo can)
  (diagonA p lmo nmo can)
  (carcel p lmo nmo)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


;*****************************************************************************
;* * * Generación de las cargas de cualquier celosia
;*****************************************************************************

(defun carcel (p lmo nmo / sup dis mod p1)

  (cprev)
  (setq  sup (getreal "\nCarga superficial de la cubierta (T/m2)")
         dis (getdist "\nDistancia entre cerchas paralelas (m)")
         p1  (list (car p) (+ (cadr p) can))
         mod (* (* sup dis) lmo)
  )
  (repeat (+ nmo 1)
    (insfue "90" p1 mod)
    (setq p1 (polar p1 0 lmo))
  )
)
```

6.2.3 Generación de Celosías tridimensionales

```
; ************ TRAZADO DE CELOSÍAS TRIDIMENSIONALES

; cordon        GENERACION  CORDONES
; mont          GENERACIÓN  DE MONTANTES
; diag1         GENERACION  DE DIAGONALES   (IDA)
; diag          GENERACION  DE DIAGONALES   (IDA Y VUELTA)
; celtri1       GENERACION  DE UNA CELOSIA  (TIPO 1)
; celtri2       GENERACION  DE UNA CELOSIA  (TIPO 2)
; celtri3       GENERACION  DE UNA CELOSIA  (TIPO 3)

; cordon     GENERACION CORDONES EN CERCHAS
; mont       GENERACIÓN DE MONTANTES EN CERCHAS TRIDIMENSIONALES
; diag1      GENERACION DE DIAGONALES EN CERCHAS TRIDIMENSIONALES (IDA)
; diag       GENERACION DE DIAGONALES EN CERCHAS TRIDIMENSIONALES (IDA Y VUELTA)
; celtri1    GENERACION DE UNA CELOSIA TRIANGULAR TRIDIMENSIONAL (TIPO 1)
; c:celtri1
; celtri2    GENERACION DE UNA CELOSIA TRIANGULAR TRIDIMENSIONAL (TIPO2)
; c:celtri2
; celtri3    GENERACION DE UNA CERCHA CUADRADA TRIDIMENSIONAL (TIPO 3)
; c:celtri3


;*****************************************************************************
;* * * GENERACION CORDONES EN CERCHAS
```

```
;*****************************************************************************

(defun cordon (p1 lmo nmo / p2)

  (cprev)

  (setq p2 (list (car p1) (+ (cadr p1) lmo) (last p1)))
  (repeat nmo
    (command "_LINE" p1 p2 "")
    (setq p1 p2
          p2 (polar p2 (/ pi 2) lmo)
    )
  )
)


;*****************************************************************************
;* * * GENERACIÓN DE MONTANTES EN CERCHAS TRIDIMENSIONALES
;*****************************************************************************

(defun mont (p1 p2 lmo nmo)

  (cprev)

  (repeat (+ nmo 1)
    (command "_LINE" p1 p2 "")
    (setq p1 (polar p1 (/ pi 2) lmo)
          p2 (polar p2 (/ pi 2) lmo)
    )
  )
)


;*****************************************************************************
;* * * GENERACION DE DIAGONALES EN CERCHAS TRIDIMENSIONALES (IDA)
;*****************************************************************************

(defun diag1 (i f lmo nmo / p1 p2)

  (cprev)

  (setq  p1 i
         p2 f
  )
  (repeat nmo
    (command "_LINE" p1 p2 "")
    (setq p1 (polar p1 (/ pi 2) lmo)
          p2 (polar p2 (/ pi 2) lmo)
    )
  )
)


;*****************************************************************************
;* * * GENERACION DE DIAGONALES EN CERCHAS TRIDIMENSIONALES (IDA Y VUELTA)
;*****************************************************************************

(defun diag (i f lmo nmo / p1 p2)

  (cprev)

  (setq  p1 i
         p2 f
  )
  (repeat nmo
    (command "_LINE" p1 p2 "")
```

```
    (setq p1 (polar p1 (/ pi 2) lmo)
          p2 (polar p2 (/ pi 2) lmo)
    )
  )
  (setq -lmo (* (~ 0) lmo)
        p1   (list (car i) (+ (cadr i) luz) (last i))
        p2   (list (car f) (+ (cadr f) (* (- nmo 1) lmo)) (last f))
  )
  (repeat nmo
    (command "_LINE" p1 p2 "")
    (setq p1 (polar p1 (/ pi 2) -lmo)
          p2 (polar p2 (/ pi 2) -lmo)
    )
  )
)



;*****************************************************************************
;* * * GENERACION DE UNA CELOSIA TRIANGULAR TRIDIMENSIONAL (TIPO 1)
;*****************************************************************************

(defun celtri1 (/ can lmo luz nmo p l)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "STR*"))
    (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01" "")
  )
  (setq p   (getpoint "\nPunto de inserci¢n de la cercha ")
        luz (getdist p "n\nLuz entre apoyos: ")
        nmo (getint "\nNumero de particiones: ")
        lmo (/ luz nmo)
        can (/ luz 15)
        prf (/ can (/ (sin 1.04719) (cos 1.04719)))
  )
                                          ;cordones
  (setq p1 p)
  (cordon p1 lmo nmo)
  (setq p1 (list (+ (car p) prf)
                 (+ (cadr p) (/ lmo 2))
                 (+ (last p) can)
           )
  )
  (cordon p1 lmo (- nmo 1))
  (setq p1 (mapcar '+ p (list (* 2 prf) 0 0)))

  (cordon p1 lmo nmo)
                                          ;diagonales
  (setq i p
        f (list  (+ (car p) prf)
                 (+ (cadr p) (/ lmo 2))
                 (+ (last p) can)
          )
  )
  (diag i f lmo nmo)
  (setq i (mapcar '+ p (list (* 2 prf) 0 0))
        f (list  (+ (car p) prf)
                 (+ (cadr p) (/ lmo 2))
```

```
                (+ (last p) can)
          )
  )
  (diag i f lmo nmo)
                                          ;montantes
  (setq  p1 p
         p2 (list (+ (car p) (* 2 prf)) (cadr p) (last p))
  )
  (mont p1 p2 lmo nmo)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:celtri1 () (celtri1))


;*****************************************************************************
;* * * GENERACION DE UNA CELOSIA TRIANGULAR TRIDIMENSIONAL (TIPO2)
;*****************************************************************************

(defun celtri2 (/ can lmo luz nmo p l)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "STR*"))
    (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01" "")
  )
  (setq  p   (getpoint "\nPunto de inserci¢n de la cercha ")
         luz (getdist p "n\nLuz entre apoyos: ")
         nmo (getint "\nNumero de particiones: ")
         lmo (/ luz nmo)
         can (/ luz 15)
         prf (/ can (/ (sin 1.04719) (cos 1.04719)))
  )
                                          ;cordones
  (setq p1 p)
  (cordon p1 lmo nmo)
  (setq  p1 (list (+ (car p) prf)
                  (+ (cadr p) (/ lmo 2))
                  (- (last p) can)
            )
  )
  (cordon p1 lmo (- nmo 1))
  (setq p1 (mapcar '+ p (list (* 2 prf) 0 0)))
  (cordon p1 lmo nmo)
                                          ;diagonales
  (setq  i p
         f (list  (+ (car p) prf)
                  (+ (cadr p) (/ lmo 2))
                  (- (last p) can)
           )
  )
  (diag i f lmo nmo)
  (setq  i (mapcar '+ p (list (* 2 prf) 0 0))
         f (list  (+ (car p) prf)
                  (+ (cadr p) (/ lmo 2))
                  (- (last p) can)
```

```
        )
  )
  (diag i f lmo nmo)
                                        ;montantes
  (setq p1 p
        p2 (list (+ (car p) (* 2 prf)) (cadr p) (last p))
  )
  (mont p1 p2 lmo nmo)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:celtri2 () (celtri2))


;*****************************************************************************
;* * * GENERACION DE UNA CERCHA CUADRADA TRIDIMENSIONAL (TIPO 3)
;*****************************************************************************

(defun celtri3 (/ can lmo luz nmo p l)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "STR*"))
    (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01" "")
  )
  (setq p   (getpoint "\nPunto de inserci¢n de la cercha ")
        luz (getdist p "n\nLuz entre apoyos: ")
        nmo (getint "\nNumero de particiones: ")
        can (getdist "n\nCanto de la cercha: ")
        prf (getdist "n\nAncho de la cercha: ")
        lmo (/ luz nmo)
  )
                                        ;cordones
  (setq p1 p)
  (cordon p1 lmo nmo)
  (setq p1 (mapcar '+ p (list 0 0 can)))
  (cordon p1 (/ lmo 2) (* nmo 2))
  (setq p1 (mapcar '+ p (list prf 0 0)))
  (cordon p1 lmo nmo)
  (setq p1 (mapcar '+ p (list prf 0 can)))
  (cordon p1 (/ lmo 2) (* nmo 2))
                                        ;diagonales principales
  (setq i p
        f (mapcar '+ p (list 0 (/ lmo 2) can))
  )
  (diag i f lmo nmo)
  (setq i (mapcar '+ p (list prf 0 0))
        f (mapcar '+ p (list prf (/ lmo 2) can))
  )
  (diag i f lmo nmo)
                                        ;diagonales secundarias
  (setq i p
        f (mapcar '+ p (list prf lmo 0))
  )
  (diag1 i f lmo nmo)
  (setq i (mapcar '+ p (list 0 0 can))
```

```
        f (mapcar '+ p (list prf lmo can))
  )
  (diag1 i f lmo nmo)
                                          ;montantes
  (setq  p1 p
        p2 (mapcar '+ p (list 0 0 can))
  )
  (mont p1 p2 lmo nmo)
  (setq  p1 p
        p2 (mapcar '+ p (list prf 0 0))
  )
  (mont p1 p2 lmo nmo)
  (setq  p1 (mapcar '+ p (list 0 0 can))
        p2 (mapcar '+ p (list prf 0 can))
  )
  (mont p1 p2 lmo nmo)
  (setq  p1 (mapcar '+ p (list prf 0 0))
        p2 (mapcar '+ p (list prf 0 can))
  )
  (mont p1 p2 lmo nmo)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:celtri3 () (celtri3))
```

6.2.4 Generación de Mallas Espaciales

```
; ********** MALLAS ESPACIALES (según NTE EAE)

; tipobar     PREDIMENSIONADO DE BARRAS DE UNA MALLA ESPACIAL O DE INTERVALO DE
;             MÓDULO SUGERIDO (segun NTE EAE)
; modmalla    SELECCION DEL MODULO MAS APROPIADO PARA UNA MALLA
; capasei     TRAZADO BARRAS CAPA SUPERIOR E INFERIOR DE UNA MALLA NO ALIGERADA
; capinfal    TRAZADO DE LAS BARRAS DE LA CAPA INFERIOR DE UNA MALLA ALIGERADA
; capdiaal    TRAZADO BARRAS DIAGONALES DE UNA MALLA ESPACIAL ALIGERADA
; diagmall    TRAZADO BARRAS DIAGONALES DE UNA MALLA ESPACIAL
; dimbarma    CONSTRUYE EL NOMBRE DEL TIPO DE LINEA CORRESPONDIENTE AL
;             DIMENSIONADO Y LO ASIGNA AUTOMATICAMENTE
; insfuema    INSERCION DE LAS CARGAS PUNTUALES EN UNA MALLA ESPACIAL
; malla       GENERACION AUTOMATICA DE UNA MALLA ESPACIAL  (segun NTE-EAE)



;*****************************************************************************
;* * *        PREDIMENSIONADO DE BARRAS DE UNA MALLA ESPACIAL O DE INTERVALO DE
;             MÓDULO SUGERIDO (segun NTE EAE)
;*****************************************************************************

(defun tipobar (mod lme ab cst tr q b r)

  (if (/= nil (posicion mod L))
    (progn
      (setq tr (nth techo tr))
      (if (/= nil (listp (nth 0 b)))
        (setq b (nth techo b))
      )
      (if (and (/= nil (posicion cst q)) (/= nil tr))
```

```
        (progn
          (setq tr (nth techo tr))
          (if (and (/= nil (posicion lme b)) (/= nil tr))
            (progn
              (setq tr (nth techo tr))
              (if (and (/= nil (posicion ab r)) (/= nil tr))
                (progn (setq tr (nth suelo tr)))
                (progn
                  (prompt
                    "\n Proporcion fuera de rango para la NTE-EAE. "
                  )
                  (prompt "\n Deberia haber estado entre ")
                  (prin1 (car r))
                  (prompt " y ")
                  (prin1 (last r))
                  (prompt "\n El proceso continuara sin las sugerencias "
                  )
                  (prompt "\n de la Norma Tecnológica.")
                  (setq tr nil)
                )
              )
            )
            (progn
              (prompt "\n Lado menor fuera de rango para la NTE-EAE. ")
              (prompt "\n Deberia haber estado entre ")
              (prin1 (car b))
              (prompt " y ")
              (prin1 (last b))
              (prompt "\n El proceso continuara sin las sugerencias ")
              (prompt "\n de la Norma Tecnológica.")
              (setq tr nil)
            )
          )
        )
        (progn (prompt "\n Sobrecarga fuera de rango para la NTE-EAE. ")
               (prompt "\n Deberia haber estado entre ")
               (prin1 (car q))
               (prompt " y ")
               (prin1 (last q))
               (prompt "\n El proceso continuara sin las sugerencias ")
               (prompt "\n de la Norma Tecnológica.")
               (setq tr nil)
        )
      )
    )
    (progn (prompt "\n Modulo fuera de rango para la NTE-EAE. ")
           (prompt "\n Deberia haber estado entre ")
           (prin1 (car L))
           (prompt " y ")
           (prin1 (last L))
           (prompt "\n El proceso continuara sin las sugerencias ")
           (prompt "\n de la Norma Tecnológica.")
           (setq tr nil)
    )
  )
)


;*****************************************************************************
;* * * SELECCION DEL MODULO MAS APROPIADO PARA UNA MALLA (segun NTE EAE)
;*****************************************************************************

(defun modmalla  (lma lme mi ms / m mod)

  (setq  mod mi
         m   (list 0)
```

```
  )
  (while (<= mod ms)
    (while (and  (<= mod ms)
                 (or (>= (rem lma mod) 0.01) (>= (rem lme mod) 0.01))
           )
      (setq mod (+ mod 0.01))
    )
    (setq m   (cons mod m)
          mod (+ mod 0.01)
    )
  )
  (if (> (car m) ms)
    (setq m (cdr m))
  )
  (cdr (reverse m))
)


;*****************************************************************************
;* * * TRAZADO BARRAS CAPA SUPERIOR E INFERIOR DE UNA MALLA NO ALIGERADA (NTE-EAE)
;*****************************************************************************

(defun capasei (mod pin v h / o p q l)

  (cprev)
  (setq  p pin
         o pin
  )
  (repeat h
    (setq l (strcat "@" (rtos mod 2 2) ",0.0,0.0"))
    (repeat v
      (setq q (mapcar '+ (list 0.0 mod 0.0) p))
      (command "_LINE" p q l "")
      (setq p q)
    )
    (setq p (mapcar '+ (list mod 0.0 0.0) o)
          o p
    )
  )
  (repeat v
    (setq q (mapcar '+ (list 0.0 mod 0.0) p))
    (command "_LINE" p q "")
    (setq p q)
  )
  (setq p pin)
  (repeat h
    (setq q (mapcar '+ (list mod 0.0 0.0) p))
    (command "_LINE" p q "")
    (setq p q)
  )
)


;*****************************************************************************
;* * * TRAZADO DE LAS BARRAS DE LA CAPA INFERIOR DE UNA MALLA ALIGERADA
;*****************************************************************************

(defun capinfal  (mod pin v h / p q s1 s2 o r)

  (cprev)
  (setq p pin)
  (repeat 2
    (repeat h
      (setq q (mapcar '+ (list mod 0.0 0.0) p))
      (command "_LINE" p q "")
      (setq p q)
```

```
    )
    (setq p (mapcar '+ (list 0.0 (* mod v) 0.0) pin))
  )

  (setq p pin)
  (repeat 2
    (repeat v
      (setq q (mapcar '+ (list 0.0 mod 0.0) p))
      (command "_LINE" p q "")
      (setq p q)
    )
    (setq p (mapcar '+ (list (* mod h) 0.0 0.0) pin))
  )

  (if (= (rem v 2) 0)
    (setq s1 (/ v 2)
          s2 s1
    )
    (setq s1 (/ (+ v 1) 2)
          s2 (- s1 1)
    )
  )
  (setq o (mapcar '+ (list 0.0 (* (~ 0) mod) 0.0) pin))
  (repeat s1
    (setq o (mapcar '+ (list 0.0 (* mod 2) 0.0) o)
          p o
          r 1
    )
    (repeat h
      (setq r (* (~ 0) r)
            q (mapcar '+ (list mod (* mod r) 0.0) p)
      )
      (command "_LINE" p q "")
      (setq p q)
    )
  )
  (setq o (mapcar '+ (list 0.0 (* (~ 0) mod) 0.0) pin))
  (repeat s2
    (setq o (mapcar '+ (list 0.0 (* mod 2) 0.0) o)
          p o
          r (~ 0)
    )
    (repeat h
      (setq r (* (~ 0) r)
            q (mapcar '+ (list mod (* mod r) 0.0) p)
      )
      (command "_LINE" p q "")
      (setq p q)
    )
  )
)


;*****************************************************************************
;* * * TRAZADO BARRAS DIAGONALES DE UNA MALLA ESPACIAL ALIGERADA (NTE EAE)
;*****************************************************************************

(defun capdiaal  (mod pin v h / p1 p2 p3 p4 p5 i d1 d2 d3 d4 p q d)

  (cprev)
  (setq  p1 pin
         p2 (mapcar '+ (list mod 0.0 0.0) p1)
         p3 (mapcar '+ (list 0.0 mod 0.0) p1)
         p4 (mapcar '+ (list mod mod 0.0) p1)
         p5 (mapcar '+ (list (/ mod 2) (/ mod 2) (* (~ 0) a)) p1)
         i  1
```

```
  )
  (command "_LINE" p1 p5 "")
  (setq d1 (entlast))
  (command "_LINE" p5 p4 "")
  (setq d2 (entlast))
  (command "_LINE" p2 p5 "")
  (setq d3 (entlast))
  (command "_LINE" p5 p3 "")
  (setq d4 (entlast))
  (while (< i (* h v))
    (setq i (1+ i)
          d (fix (/ (- i 0.5) h))
          p (mapcar '+ (list (* (~ 0) mod) (* d mod) 0.0) pin)
          q (mapcar '+ (list (* mod (- i (* d h))) 0.0 0.0) p)
    )
    (if (or (<= i h)
            (> i (* h (- v 1)))
            (= (- i (* d h)) 1)
            (= i (* h (1+ d)))
            (= (rem (+ (rem (- i (* d h)) 2) d) 2) 0)
        )
      (command "_COPY" d1 d2 d3 d4 "" pin q)
    )
  )
)


;*****************************************************************************
;* * * TRAZADO BARRAS DIAGONALES DE UNA MALLA ESPACIAL (segun NTE-EAE)
;*****************************************************************************

(defun diagmall  (mod pin v h)

  (cprev)
  (setq  p1 pin
         p2 (mapcar '+ (list mod 0.0 0.0) p1)
         p3 (mapcar '+ (list 0.0 mod 0.0) p1)
         p4 (mapcar '+ (list mod mod 0.0) p1)
         p5 (mapcar '+ (list (/ mod 2) (/ mod 2) (* (~ 0) a)) p1)
  )
  (command "_LINE" p1 p5 "")
  (setq d1 (entlast))
  (command "_LINE" p5 p4 "")
  (setq d2 (entlast))
  (command "_LINE" p2 p5 "")
  (setq d3 (entlast))

  (command "_LINE" p5 p3 "")
  (setq d4 (entlast))
  (command "_ARRAY" d1 d2 d3 d4 "" "R" v h mod mod)
)



;*****************************************************************************
;* * * CONSTRUYE EL NOMBRE DEL TIPO DE LINEA CORRESPONDIENTE AL
;       DIMENSIONADO Y LO ASIGNA AUTOMATICAMENTE
;*****************************************************************************

(defun dimbarma  (dim / i nom_mat mat m num_mat p1 p2 tl)

                                        ; BÚSQUEDA DEL NÚMERO DE MATERIAL
                                        ; CORRESPONDIENTE AL ACERO
  (progn (setq i 0
               nom_mat ""
         )
```

```
        (while (= nil (wcmatch nom_mat "ACERO"))
          (setq mat      (ssname (ssget "X" (list (cons 2 "MATERIAL"))) i)
                m        (entnext mat)
                nom_mat (cdr (assoc 1 (entget m)))
                i        (1+ i)
          )
        )
  )
                                        ;  CREACIÓN DEL NOMBRE DEL TIPO DE
                                        ;  LÍNEA CORRESPONDIENTE AL DIMENSIONADO

  (setq num_mat (cdr (assoc 1 (entget (entnext m))))
        mat     (chr (+ 64 (atoi num_mat)))
        p1      (rtos (fix dim) 2 0)
        p2      (substr (rtos dim 2 1) (strlen (rtos dim 2 1)))
        tl      (strcat "F" mat "0" "I" p1 "I" p2)
  )
                                        ;  CREACIÓN Y CARGA DEL NUEVO TIPO LINEA

  (if (= nil (tblsearch "LTYPE" tl))
    (progn (creaTl tl)
           (cargaTl tl)
    )
  )
  (setq fich (open "c:/cid/cad/st.lin" "w"))
  (close fich)
  (setvar "CELTYPE" tl)

)


;*****************************************************************************
;* * * INSERCION DE LAS CARGAS PUNTUALES EN UNA MALLA ESPACIAL SEGUN NTE-EAE
;*****************************************************************************

(defun insfuema  (mod pin v h sob / m i d p r s)

  (cprev)
  (setq m (* mod mod (/ sob 1000))
        i 0
  )
  (while (< i (* (1+ h) (1+ v)))
    (setq i (1+ i)
          d (fix (/ (- i 0.5) (1+ h)))
          p (mapcar '+
                    (list (* mod (1- (- i (* d (1+ h))))) (* mod d) 0.0)
                    pin
            )


    )
    (cond ((or (= i 1)
               (= i (1+ h))
               (= i (1+ (* (1+ h) v)))
               (= i (* (1+ h) (1+ v)))
           )
           (progn (setq p (trans p 1 0))
                  (command "_UCS" "X" 90)
                  (setq  p (trans p 0 1)
                         r (mapcar '+ (list 0.0 0.1 0.0) p)
                         s (mapcar '+ (list 0.0 (/ m 4) 0.0) p)
                  )
                  (command "_PLINE" p "_W" 0 0.05 r "_W" 0 0 s "")
                  (command "_UCS" "X" -90)
           )
          )
```

```
          ((or (< i (1+ h))
               (> i (* (1+ h) v))
               (= (- i (* d (1+ h))) 1)
               (= i (* (1+ d) (1+ h)))
           )
           (progn (setq p (trans p 1 0))
                  (command "SCP" "N" "X" 90)
                  (setq  p (trans p 0 1)
                         r (mapcar '+ (list 0.0 0.1 0.0) p)
                         s (mapcar '+ (list 0.0 (/ m 2) 0.0) p)
                  )
                  (command "_PLINE" p "_W" 0 0.05 r "_W" 0 0 s "")
                  (command "SCP" "N" "X" -90)
           )
          )
          (((= nil i)
           (progn (setq p (trans p 1 0))
                  (command "SCP" "N" "X" 90)
                  (setq  p (trans p 0 1)
                         r (mapcar '+ (list 0.0 0.1 0.0) p)
                         s (mapcar '+ (list 0.0 m 0.0) p)
                  )
                  (command "_PLINE" p "_W" 0 0.05 r "_W" 0 0 s "")
                  (command "SCP" "N" "X" -90)
           )
          )
    )
  )
)


;*********************************************************************
;* * * GENERACION AUTOMATICA DE UNA MALLA ESPACIAL  (segun NTE-EAE)
;*********************************************************************

(defun malla (tabla /      pin   p1    lma   p2    ori
              lme   pro    ab    sob   L     lm    apo   niv   h
              pp    alig   q1    b1    ab1   q2    b2    ab2   mt
              ma    ts     ti    tdt   tda   cst   m     td    NTE_mod
              lh    mod    a     s     i     d     v     NTE_dim
              Ace   dim_nom      man   Ace_dif     tl    p     lc
             )

;  ADECUACIÓN DE VARIABLES DE SISTEMA

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (setvar "ORTHOMODE" 0)
  (setvar "OSMODE" 47)
  (scpu)


;  CREACIÓN DE LAS CAPAS NECESARIAS PARA EL TRAZADO

  (if (= (tblsearch "layer" "STR_CSUP") nil)
    (command "_LAYER" "_N" "STR_CSUP" "_COLOR" "4" "STR_CSUP" "")
  )
  (if (= (tblsearch "layer" "STR_CINF") nil)
    (command "_LAYER" "_N" "STR_CINF" "_COLOR" "3" "STR_CINF" "")
  )
  (if (= (tblsearch "layer" "STR_DIAG") nil)
    (command "_LAYER" "_N" "STR_DIAG" "_COLOR" "5" "STR_DIAG" "")
  )
```

```
;  INTRODUCCIÓN DE LOS PARAMETROS GEOMETRICOS GENERALES DE LA MALLA

  (setq  pin (getpoint "\n Punto de insercion de la malla: ")
         p1  (getpoint pin
                       "\n Punto que determina el lado mayor de la malla: "
             )
         lma (distance pin p1)
  )
  (command "_LINE" pin p1 "")
  (SCPEntidad)
  (entdel (entlast))
  (setvar "ORTHOMODE" 1)
  (setq  pin (trans pin 0 1)
         p2  (getpoint pin
                       "\n Punto que determina el lado menor de la malla: "
             )
  )
  (while (>= (abs (car p2)) 0.01)
    (prompt
      "\n ¡ATENCION! El punto que determina el lado menor, "
    )
    (prompt
      "\n            no puede encontrarse sobre el lado mayor. "
    )
    (setq p2 (getpoint pin "\n Intentelo de nuevo: "))
  )
  (if (> (cadr p2) 0)
    (setq ori 1)
    (setq ori -1)
  )
  (setq lme (distance pin p2))
  (if (> lme lma)
    (progn
      (prompt
        "\n ¡ATENCION! La dimension introducida como lado menor, "
      )
      (prompt "\n            es superior a la del lado mayor. ")
      (prompt
        "\n El programa corregira estos datos, intercambiando los valores."
      )
      (setq L    lme
            lme  lma
            lma  L
            pro  -1
      )
    )
    (setq pro 1)
  )
  (if (= (* pro ori) -1)
    (command "_UCS" "_O" p2)
  )
  (if (= pro -1)
    (command "_UCS" "Z" "270")
  )
  (setq  ab  (/ lma lme)
         sob (getreal
               "\n Introduzca el valor de (Concarga + Uso + Nieve) sin mayorar
(kg/m2): "
             )
         L   (list 2.0 2.5 3.0 3.5 4.0)
         lm  (list (list 2.0 2.5)
                   (list 2.5 3.0)
                   (list 2.5 3.5)
                   (list 2.5 4.0)
```

```
                (list 3.0 3.5)
                (list 3.0 4.0)
                (list 3.5 4.0)
                (list 4.0 4.0)
          )
  )

;  INTRODUCCIÓN DE LOS DATOS REFERENTES AL APOYO

  (while (and (/= apo "TO")
              (/= apo "To")
              (/= apo "tO")
              (/= apo "to")
              (/= apo "AL")
              (/= apo "Al")
              (/= apo "aL")
              (/= apo "al")
         )
    (prompt "\n ¿Hay apoyo en todos los nudos del perimetro ")
    (prompt
      "\n  o solo en los nudos alternos del mismo? (To/Al) "
    )
    (setq apo (getstring))
  )
  (while (and (/= niv "SUP")
              (/= niv "Sup")
              (/= niv "sUP")
              (/= niv "sup")
              (/= niv "INF")
              (/= niv "Inf")
              (/= niv "iNF")
              (/= niv "inf")
         )
    (prompt "\n ¿Se apoya la malla sobre la trama superior ")
    (prompt "\n  o sobre la trama inferior? (Sup/Inf) ")
    (setq niv (getstring))
  )

;  CARGA DE LOS PARAMETROS CORRESPONDIENTES AL TIPO DE MALLA

  (cond ((or (wcmatch tabla "cl45") (wcmatch tabla "CL45")) (cl45))
        ((or (wcmatch tabla "cl55") (wcmatch tabla "CL55")) (cl55))
        ((or (wcmatch tabla "c45") (wcmatch tabla "C45")) (c45))
        ((or (wcmatch tabla "c55") (wcmatch tabla "C55")) (c55))
  )

;  DISCRIMINACION DE PARAMETROS EN FUNCION DE LOS APOYOS

  (if (or (= apo "TO") (= apo "To") (= apo "tO") (= apo "to"))
    (setq cst (+ (car pp) sob)
          m   mt
          td  tdt
    )
    (setq cst (+ (last pp) sob)
          m   ma
          td  tda
    )
  )

;  DETERMINACION DEL MODULO

  (while (and (/= NTE_mod "N")
```

```
              (/= NTE_mod "n")
              (/= NTE_mod "S")
              (/= NTE_mod "s")
          )
    (prompt
      "\n ¿Desea que el programa le proponga modulos de acuerdo con la NTE-EAE?
(S/N)"
    )
    (setq NTE_mod (getstring))
  )
  (if (or (wcmatch NTE_mod "S") (wcmatch NTE_mod "s"))
    (progn
      (setq lh (tipobar 2 lme ab sob m q1 b1 ab1))
      (if (and (/= nil lh) (/= 0 lh))
        (progn
          (setq  lh  (nth (- lh 1) lm)
                 mod (car (modmalla lma lme (car lh) (last lh)))
          )
          (if (/= (car lh) (last lh))
            (progn
              (prompt "\n La NTE-EAE recomienda un modulo entre ")
              (prin1 (car lh))
              (prompt " y ")
              (prin1 (last lh))
              (prompt " metros. ")
              (if (/= mod nil)
                (progn
                  (prompt "\n Dada la longitud de los lados, el modulo")
                  (prompt " mas apropiado seria de ")
                  (prin1 mod)
                  (prompt " metros. ")
                )
                (progn
                  (prompt
                    "\n Las dimensiones de los lados son tales que no hay ninguna"
                  )
                  (prompt
                    "\n dimension dentro del intervalo aconsejado que se adecue a
ambas."
                  )
                  (setq  NTE_mod  "N"
                         mod nil
                  )
                )
              )
            )
            (progn
              (prompt
                "\n Dadas las caracteristicas de la malla la NTE-EAE"
              )
              (prompt "\n sugiere como modulo mas adecuado ")
              (prin1 (car lh))
              (prompt " metros. ")
            )
          )
        )
        (setq NTE_mod "N")
      )
    )
  )
  (if (= 0 lh)
    (prompt
      "\n Combinación métrica no contemplada en la NTE-EAE-86."
    )
  )
  (if (or (wcmatch NTE_mod "N") (wcmatch NTE_mod "n"))
```

```
    (progn (prompt "\n La longitud del lado mayor introducida es de ")
           (prin1 lma)
           (prompt " metros")
           (prompt " y la del lado menor de ")
           (prin1 lme)
           (prompt " metros. ")
    )
  )
  (setq mod (getreal "\n ¿Que longitud del modulo va a tomarse? (m) "))
  (while (= nil mod)
    (prompt
      "\n ¡ATENCION! Debe introducir alguna longitud para el modulo."
    )
    (setq
      mod (getreal "\n ¿Que longitud del modulo va a tomarse? (m) ")
    )
  )
  (while (= 0 mod)
    (prompt
      "\n ¡ATENCION! La longitud del modulo debe ser distinta de 0."
    )
    (setq
      mod (getreal "\n ¿Que longitud del modulo va a tomarse? (m) ")
    )
  )

  (while (> 0 mod)
    (prompt
      "\n ¡ATENCION! La longitud del modulo debe ser una longitud positiva."
    )
    (setq
      mod (getreal "\n ¿Que longitud del modulo va a tomarse? (m) ")
    )
  )
  (if (/= nil mod)
    (setq a (* mod h))
    (if (/= nil lh)
      (setq mod  (car lh)
            a    (* mod h)
      )
      (setq mod  1.0
            a    (* mod h)
      )
    )
  )
  (prompt "\n La altura de la malla espacial sera de ")
  (prin1 a)
  (prompt " metros. ")

;  DETERMINACION DEL DIMENSIONADO DE LAS BARRAS SEGUN NTE-EAE

  (if (or (= NTE_mod "S") (= NTE_mod "s"))
    (progn
      (prompt
        "\n La NTE-EAE propone dimensionar las barras de las "
      )
      (prompt
        "\n mallas espaciales con perfiles tubulares huecos. "
      )

      (while (and (/= NTE_dim "S")
                  (/= NTE_dim "s")
                  (/= NTE_dim "N")
                  (/= NTE_dim "n")
             )
        (prompt
```

```
          "\n ¿Desea utilizar este tipo de perfiles para dimensionar su malla? (S/N)
"
        )
        (setq NTE_dim (getstring))
      )
    )
  )

  (if (or (= NTE_dim "S") (= NTE_dim "s"))
    (progn
      (setq s (tipobar mod lme ab cst ts q2 b2 ab2))
      (if (and (/= nil s) (/= 0 s))
        (progn
          (setq i (tipobar 2 lme ab cst ti q2 b3 ab2))
          (if (and (/= nil i) (/= 0 i))
            (progn
              (setq d (tipobar mod lme ab cst td q2 b2 ab1))
              (if (or (= nil d) (= 0 d))
                (progn
                  (prompt
                    "\n No puede sugerirse un dimensionado para las barras
diagonales, según la NTE-EAE."
                  )
                  (setq NTE_dim "N")
                )
              )
            )

            (progn
              (prompt
                "\n No puede sugerirse un dimensionado para las barras inferiores,
según la NTE-EAE."
              )
              (setq NTE_dim "N")
            )
          )
        )
        (progn
          (prompt
            "\n No puede sugerirse un dimensionado para las barras superiores, según
la NTE-EAE."
          )
          (setq NTE_dim "N")
        )
      )
    )
  )
  (if (or (= NTE_dim "N") (= NTE_dim "n"))
    (progn
      (prompt
        "\n Por decision propia o porque alguno de los parametros propuestos "
      )
      (prompt
        "\n para la malla escapan a los contemplados en la NTE-EAE, "
      )
      (prompt
        "\n el programa la construira sin predimensionar las barras. "
      )
      (prompt
        "\n Tras el trazado deberan dimensionarse con el tipo de perfil deseado. "
      )
      (setq dim "N")
    )
  )
```

```
;  ELECCION DEL TIPO DE ACERO


  (if (or (= NTE_dim "S") (= NTE_dim "s"))
    (progn
      (while (and (/= Ace 37) (/= Ace 42))
        (setq Ace
               (getreal
                  "\n ¿Qué tipo de acero utilizará para la malla, A-37 ó A-42b?
(37/42) "
               )
        )
      )
      (if (and (= Ace 37) (or (= s 17) (= i 17) (= d 17)))
        (progn
          (prompt
            "\n El predimensionado con acero A-37 resulta insuficiente en algún caso"
          )
          (prompt
            "\n según la NTE-EAE. El programa cambiará automáticamente a acero A-42b.
"
          )
          (setq Ace 42)
        )
      )
      (if (= Ace 37)
        (setq Ace_dif 0)
        (setq Ace_dif 1)
      )


;  COMUNICACIÓN DE DIMENSIONADOS Y POSIBILIDAD DE MODIFICACION (BARRAS SUPERIORES)

      (setq dim_nom (list 40.2      50.2     60.2     70.2     80.2
                          70.3      80.3     90.3     100.3    100.4
                          125.4     125.5    155.5    175.5    200.5
                          200.6     200.8
                         )
            s       (nth (- s Ace_dif) dim_nom)
      )
      (progn
        (prompt
          "\n La NTE-EAE propone para las barras superiores, perfiles tubulares D "
        )
        (prin1 s)
        (prompt " ,")
        (while
          (and (/= man "S") (/= man "s") (/= man "N") (/= man "n"))
           (setq man (getstring
                       "\n ¿Desea mantener este predimensionado? (S/N)"
                     )
           )
        )
        (if (or (wcmatch man "N") (wcmatch man "n"))
          (progn
            (setq s nil)
            (while (= nil (member s dim_nom))
              (prompt
                "\n ¿Cuál de los siguientes perfiles desea utilizar? "
              )
              (if (= Ace 42)
                (prompt
                  "\n 40.2  50.2  60.2  70.2  80.2  70.3  80.3  90.3  100.3 "
                )
                (prompt
                  "\n 50.2  60.2  70.2  80.2  70.3  80.3  90.3  100.3 "
```

```
            )
          )
          (prompt
            "\n 100.4  125.4  125.5  155.5  175.5  200.5  200.6  200.8 "
          )
          (setq s (getreal))
        )
      )
    )
    (setq man nil)
  )

;  COMUNICACIÓN DE DIMENSIONADOS Y POSIBILIDAD DE MODIFICACION (BARRAS INFERIORES)

  (setq i (nth (- i Ace_dif) dim_nom))
  (progn
    (prompt
      "\n La NTE-EAE propone para las barras inferiores, perfiles tubulares D "
    )
    (prin1 i)
    (prompt " ,")
    (while
      (and (/= man "S") (/= man "s") (/= man "N") (/= man "n"))
       (setq man (getstring
                   "\n ¿Desea mantener este predimensionado? (S/N)"
                 )
       )
    )
    (if (or (wcmatch man "N") (wcmatch man "n"))
      (progn
        (setq i nil)
        (while (= nil (member i dim_nom))
          (prompt
            "\n ¿Cuál de los siguientes perfiles desea utilizar? "
          )
          (if (= Ace 42)
            (prompt
              "\n 40.2  50.2  60.2  70.2  80.2  70.3  80.3  90.3  100.3 "
            )
            (prompt
              "\n 50.2  60.2  70.2  80.2  70.3  80.3  90.3  100.3 "
            )
          )
          (prompt
            "\n 100.4  125.4  125.5  155.5  175.5  200.5  200.6  200.8 "
          )
          (setq i (getreal))
        )
      )
    )

    (setq man nil)
  )

;  COMUNICACIÓN DE DIMENSIONADOS Y POSIBILIDAD DE MODIFICACION (BARRAS DIAGONALES)

  (setq d (nth (- d Ace_dif) dim_nom))
  (progn
    (prompt
      "\n La NTE-EAE propone para las barras diagonales, perfiles tubulares D "
    )
    (prin1 d)
    (prompt " ,")
    (while
```

```
        (and (/= man "S") (/= man "s") (/= man "N") (/= man "n"))
         (setq man (getstring
                     "\n ¿Desea mantener este predimensionado? (S/N)"
                   )
         )
       )
       (if (or (wcmatch man "N") (wcmatch man "n"))
         (progn
           (setq d nil)
           (while (= nil (member d dim_nom))
             (prompt
               "\n ¿Cuál de los siguientes perfiles desea utilizar? "
             )
             (if (= Ace 42)
               (prompt
                 "\n 40.2  50.2  60.2  70.2  80.2  70.3  80.3  90.3  100.3 "
               )
               (prompt
                 "\n 50.2  60.2  70.2  80.2  70.3  80.3  90.3  100.3 "
               )
             )
             (prompt
               "\n 100.4  125.4  125.5  155.5  175.5  200.5  200.6  200.8 "
             )
             (setq d (getreal))
           )
         )
       )
       (setq man nil)
     )
     (prompt "\nLos dimensionados escogidos son D ")
     (prin1 s)
     (prompt "\ para las barras superiores, D ")
     (prin1 i)
     (prompt "\ para las barras inferiores y D ")
     (prin1 d)
     (prompt "\ para las barras diagonales. ")

   )
)
(terpri)

                                        ;  VACIADO DE VARIABLES INNECESARIAS

(setq  p1 nil
       p2 nil
       ori nil
       pro nil
       ab nil
       L nil
       lm nil
       apo nil
       pp nil
       q1 nil
       b1 nil
       ab1 nil
       q2 nil
       b2 nil
       ab2 nil
       mt nil
       ma nil
       ts nil
       ti nil
       tdt nil
       tda nil
       cst nil
```

```
       m nil
       td nil
       lh nil
       NTE_mod  nil
       dim_nom  nil
       Ace_dif  nil
)
                        ; CALCULO DE PARAMETROS PARA EL TRAZADO DE LA MALLA

(setvar "OSMODE" 0)
(setq  h   (fix (/ (+ lma 0.01) mod))
       v   (fix (/ (+ lme 0.01) mod))
       lma nil
       lme nil
)
(prompt "\n El numero de modulos horizontales es de ")
(if (or (= niv "INF") (= niv "Inf") (= niv "iNF") (= niv "inf"))
  (prin1 (+ h 1))
  (prin1 h)
)
(prompt "\ y el de verticales de ")
(if (or (= niv "INF") (= niv "Inf") (= niv "iNF") (= niv "inf"))
  (prin1 (+ v 1))
  (prin1 v)
)
(setq l (getvar "CLAYER"))


(setvar "UCSICON" 0)                    ; Desactiva el símbolo del SCP

                                        ;  TRAZADO DE LA TRAMA SUPERIOR

(setq tl (getvar "CELTYPE"))
(if (or (= NTE_dim "S") (= NTE_dim "s"))
  (dimbarma s)
)
(if (not (wcmatch l "STR_CSUP"))
  (command "_LAYER" "_S" "STR_CSUP" "")
)
(if (or (= niv "INF") (= niv "Inf") (= niv "iNF") (= niv "inf"))
  (progn (setq p (mapcar '+
                          (list (* (~ 0) (/ mod 2)) (* (~ 0) (/ mod 2)) a)
                          pin
                 )
         )
         (capasei mod p (+ v 1) (+ h 1))
  )
  (capasei mod pin v h)
)
                                        ;  TRAZADO DE LA TRAMA INFERIOR

(if (or (= NTE_dim "S") (= NTE_dim "s"))
  (dimbarma i)
)
(command "_LAYER" "_S" "STR_CINF" "")
(cond
  ((and (= alig "NO")
        (or (= niv "SUP") (= niv "Sup") (= niv "sUP") (= niv "sup"))
   )
   (progn (setq p (mapcar '+ (list (/ mod 2) (/ mod 2) (* (~ 0) a)) pin))
          (capasei mod p (- v 1) (- h 1))
   )
  )
  ((and (= alig "NO")
        (or (= niv "INF") (= niv "Inf") (= niv "iNF") (= niv "inf"))
   )
```

```
   (capasei mod pin v h)
  )
  ((and (= alig "SI")
        (or (= niv "SUP") (= niv "Sup") (= niv "sUP") (= niv "sup"))
   )
   (progn (setq p (mapcar '+ (list (/ mod 2) (/ mod 2) (* (~ 0) a)) pin))
          (capinfal mod p (- v 1) (- h 1))
   )
  )
  ((and (= alig "SI")
        (or (= niv "INF") (= niv "Inf") (= niv "iNF") (= niv "inf"))
   )
   (capinfal mod pin v h)
  )
)
                                          ;  TRAZADO DE LA TRAMA DIAGONAL

(if (or (= NTE_dim "S") (= NTE_dim "s"))
  (dimbarma d)
)
(command "_LAYER" "_S" "STR_DIAG" "")
(cond
  ((and (= alig "NO")
        (or (= niv "SUP") (= niv "Sup") (= niv "sUP") (= niv "sup"))
   )
   (diagmall mod pin v h)
  )
  ((and (= alig "NO")
        (or (= niv "INF") (= niv "Inf") (= niv "iNF") (= niv "inf"))
   )
   (progn (setq
            p (mapcar '+
                      (list (* (~ 0) (/ mod 2)) (* (~ 0) (/ mod 2)) a)
                      pin
              )
          )
          (diagmall mod p (+ v 1) (+ h 1))
   )
  )
  ((and (= alig "SI")
        (or (= niv "SUP") (= niv "Sup") (= niv "sUP") (= niv "sup"))
   )
   (capdiaal mod pin v h)
  )
  ((and (= alig "SI")
        (or (= niv "INF") (= niv "Inf") (= niv "iNF") (= niv "inf"))
   )
   (progn (setq
            p (mapcar '+
                      (list (* (~ 0) (/ mod 2)) (* (~ 0) (/ mod 2)) a)
                      pin
              )
          )
          (capdiaal mod p (+ v 1) (+ h 1))
   )
  )
)
(setvar "CELTYPE" tl)
                                          ;  INSERCIÓN DE LAS CARGAS EN LA MALLA

(if (= (tblsearch "layer" "HIP01") nil)
  (command "_LAYER" "_N" "HIP01" "_COLOR" "1" "HIP01" "")
)
(setq  lc (getstring
            "\n ¿En que capa de hipotesis desea colocar las cargas? "
          )
```

```
  )
  (while (= (tblsearch "layer" lc) nil)
    (prompt "\n ¡ATENCION! La capa propuesta no existe. ")
    (prompt
      "\n ¿En que capa de las existentes desea colocar las cargas? "
    )
    (setq lc (getstring))
  )
  (command "_LAYER" "_S" lc "")
  (if (= nil ef)
    (setq ef 1)
  )
  (cond ((or (= niv "SUP") (= niv "Sup") (= niv "sUP") (= niv "sup"))
         (insfuema mod pin v h sob)
        )
        ((or (= niv "INF") (= niv "Inf") (= niv "iNF") (= niv "inf"))
         (progn (setq
                  p (mapcar '+
                            (list (* (~ 0) (/ mod 2)) (* (~ 0) (/ mod 2)) a)
                            pin
                    )
                )
                (insfuema mod p (1+ v) (1+ h) sob)
         )
        )
  )

  (gc)
  (scpu)
  (setvar "CLAYER" l)
  (setvar "UCSICON" 1)                    ; Activa el símbolo del SCP

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)

  (princ)                                 ; Termina en silencio
)
```

6.2.5 Generación de Sistemas Estructurales desarrollados sobre Superficies

```
; ************* SISTEMAS ESTRUCTURALES DESARROLLADOS SOBRE SUPERFICIES

;               COMPUESTOS POR BARRAS
;
; phip          Dibuja una retícula de barras dentro de un cuadrilátero alabeado
; cuadril
; C:phip
;
;               COMPUESTOS POR ELEMENTOS FINITOS 2D
;
; utira         Dibuja una tira de 3Dcaras de 3 lados
; ph3d          Dibuja elementos triangulares dentro de un cuadrilátero alabeado
; C:ph3d
; utira4        Dibuja una tira de 3Dcaras de 4 lados
; omple4d       Llena un cuadrilátero dado por cuatro puntos y con 3DCARA
; ph4d          Dibuja elementos cuadriláteros dentro de un cuadrilátero alabeado
```

```
; C:ph4d
; mur           Dibuja un muro de una altura dada con 3Dcaras de 4 lados
; c:mur
; m4            Rellena una zona delimitada por cuatro entidades
;               con 3Dcara de 4 lados
; c:m4


;*****************************************************************************
;* * * DIBUJA UNA RETICULA DE BARRAS APOYADA SOBRE UN PARABOLOIDE HIPERBOLICO
;*****************************************************************************

(defun phip (/    p1  p2  p3  p4   x1  x2  y1  y2  z1  z2  x11 y11 z11
             xn   yn  zn  l   n    nl1 nl2 poi pf   x   y   z   nr  x22
             y22 z22
            )

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq p1  (getpoint "\nPrimera esquina:")
        p2  (getpoint p1 "\nSegunda esquina:")
        nl1 (getint "\nNumero barras en lado 1-2:")
        p3  (getpoint p2 "\nTercera  esquina:")
        nl2 (getint "\nNumero barras en ese lado 2-3:")
        p4  (getpoint p3 "\nCuarta  esquina:")
  )
  (r_non)                                   ;..... direccion paralela a 1-2

  (setq x1  (/ (- (car p4) (car p1)) nl2)
        y1  (/ (- (cadr p4) (cadr p1)) nl2)
        z1  (/ (- (caddr p4) (caddr p1)) nl2)
        x2  (/ (- (car p3) (car p2)) nl2)
        y2  (/ (- (cadr p3) (cadr p2)) nl2)
        z2  (/ (- (caddr p3) (caddr p2)) nl2)
        poi p1
        pf  p2
        x11 (car p1)
        y11 (cadr p1)
        z11 (caddr p1)
        x22 (car p2)
        y22 (cadr p2)
        z22 (caddr p2)
        nr  (+ nl2 1)
  )                                         ;.... alineaciones paralelas a 1-2
  (repeat nr
    (setq xn (/ (- (car pf) (car poi)) nl1)
          yn (/ (- (cadr pf) (cadr poi)) nl1)
          zn (/ (- (caddr pf) (caddr poi)) nl1)
          x  (+ xn (car poi))
          y  (+ yn (cadr poi))
          z  (+ zn (caddr poi))
    )
    (repeat nl1
      (setq pf (list x y z))
      (command "_LINE" poi pf "")
      (setq poi  pf
            x    (+ x xn)
            y    (+ y yn)
            z    (+ z zn)
      )
    )
```

```
    (setq x11 (+ x1 x11)
          y11 (+ y1 y11)
          z11 (+ z1 z11)
          poi (list x11 y11 z11)
          x22 (+ x2 x22)
          y22 (+ y2 y22)
          z22 (+ z2 z22)
          pf  (list x22 y22 z22)
    )
  )                                        ;......... direccion paralela a 2-3

  (setq x1  (/ (- (car p2) (car p1)) nl1)
        y1  (/ (- (cadr p2) (cadr p1)) nl1)
        z1  (/ (- (caddr p2) (caddr p1)) nl1)
        x2  (/ (- (car p3) (car p4)) nl1)
        y2  (/ (- (cadr p3) (cadr p4)) nl1)
        z2  (/ (- (caddr p3) (caddr p4)) nl1)
        poi p1
        pf  p4
        x11 (car p1)
        y11 (cadr p1)
        z11 (caddr p1)
        x22 (car p4)
        y22 (cadr p4)
        z22 (caddr p4)
        nr  (+ nl1 1)
  )
  (repeat nr
    (setq xn (/ (- (car pf) (car poi)) nl2)
          yn (/ (- (cadr pf) (cadr poi)) nl2)
          zn (/ (- (caddr pf) (caddr poi)) nl2)
          x  (+ xn (car poi))
          y  (+ yn (cadr poi))
          z  (+ zn (caddr poi))
    )
    (repeat nl2
      (setq pf (list x y z))
      (command "_LINE" poi pf "")
      (setq poi  pf
            x    (+ x xn)
            y    (+ y yn)
            z    (+ z zn)
      )
    )
    (setq x11 (+ x1 x11)
          y11 (+ y1 y11)
          z11 (+ z1 z11)
          poi (list x11 y11 z11)
          x22 (+ x2 x22)
          y22 (+ y2 y22)
          z22 (+ z2 z22)
          pf  (list x22 y22 z22)
    )
  )

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun cuadril () (phip))

(defun C:phip () (phip))
```

```
;*****************************************************************************
;* * *     FUNCION PARA DIBUJAR UNA TIRA DE 3DCARA DE 3 LADOS
;*****************************************************************************

(defun utira (p1 p2 p3 p4 nel
              /
              xn yn zn poi pj pk pk1 x y z xxn yyn zzn x1 y1 z1)

  (setq xn  (/ (- (car p2) (car p1)) nel)
        yn  (/ (- (cadr p2) (cadr p1)) nel)
        zn  (/ (- (caddr p2) (caddr p1)) nel)
        xxn (/ (- (car p3) (car p4)) nel)
        yyn (/ (- (cadr p3) (cadr p4)) nel)
        zzn (/ (- (caddr p3) (caddr p4)) nel)
        poi p1
        pk1 p4
        x   (+ xn (car poi))
        y   (+ yn (cadr poi))
        z   (+ zn (caddr poi))
        pj  (list x y z)
        x1  (+ (/ xxn 2) (car pk1))
        y1  (+ (/ yyn 2) (cadr pk1))
        z1  (+ (/ zzn 2) (caddr pk1))
        pk  (list x1 y1 z1)
  )
  (repeat nel
    (3_CARA poi pj pk pk)
    (3_CARA poi pk pk1 pk1)
    (setq poi pj
          pk1 pk
          x   (+ x xn)
          y   (+ y yn)
          z   (+ z zn)
          pj  (list x y z)
          x1  (+ x1 xxn)
          y1  (+ y1 yyn)
          z1  (+ z1 zzn)
          pk  (list x1 y1 z1)
    )
  )
  (setq x1 (- x1 (/ xxn 2))
        y1 (- y1 (/ yyn 2))
        z1 (- z1 (/ zzn 2))
        pj (list x1 y1 z1)
  )
  (3_CARA poi pj pk1 pk1)
)


;*****************************************************************************
;* * *  DIBUJA ELEMENTOS 3Dcara TRIANGULARES EN CUADRILATERO ALABEADO
;*****************************************************************************

(defun ph3d (/ p1 p2 p3  p4 xn yn zn xd yd zd n np nl1 nl2 dd p11 p22 p33
               p44 x y z)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq n  -1
        np -1
        p1 (getpoint "\nPrimera esquina:")
```

```
        p2 (getpoint p1 "\nSegunda esquina:")
        dd (distance p1 p2)
  )
  (princ "\n Longitud del lado 1º= ")
  (princ dd)
  (setq  nl1 (getint "\n Numero elementos en lado 1-2:")
        p3  (getpoint p2 "\n Tercera  esquina:")
        dd  (distance p2 p3)
  )
  (princ "\n Longitud del lado 2º= ")
  (princ dd)
  (setq  nl2 (getint "\n Numero elementos en ese lado 2-3:")
        p4  (getpoint p3 "\n Cuarta  esquina:")
  )
  (r_non)
  (setq  xn  (/ (- (car p4) (car p1)) nl2)
        yn  (/ (- (cadr p4) (cadr p1)) nl2)
        zn  (/ (- (caddr p4) (caddr p1)) nl2)
        xd  (/ (- (car p3) (car p2)) nl2)
        yd  (/ (- (cadr p3) (cadr p2)) nl2)
        zd  (/ (- (caddr p3) (caddr p2)) nl2)
        p11 p1
        p22 p2
  )
  (repeat nl2
    (setq x   (+ xn (car p11))
          y   (+ yn (cadr p11))
          z   (+ zn (caddr p11))
          p44 (list x y z)
          x   (+ xd (car p22))
          y   (+ yd (cadr p22))
          z   (+ zd (caddr p22))
          p33 (list x y z)
          n   (* n np)
    )
    (if (= n 1)
      (utira p11 p22 p33 p44 nl1)
    )
    (if (= n -1)
      (utira p33 p44 p11 p22 nl1)
    )
    (setq p11 p44
          p22 p33
    )
  )

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun C:ph3d () (ph3d))


;*****************************************************************************
;* * *     FUNCION PARA DIBUJAR UNA TIRA DE 3DCARA DE 4 LADOS
;*****************************************************************************

(defun utira4 (p1 p2 p3  p4 nel
               /
               xn yn zn poi pj pk pk1 x y z xxn yyn zzn x1 y1 z1)

  (setq  xn  (/ (- (car p2) (car p1)) nel)
        yn  (/ (- (cadr p2) (cadr p1)) nel)
        zn  (/ (- (caddr p2) (caddr p1)) nel)
```

```
        xxn (/ (- (car p3) (car p4)) nel)
        yyn (/ (- (cadr p3) (cadr p4)) nel)
        zzn (/ (- (caddr p3) (caddr p4)) nel)
        poi p1
        pk1 p4
        x   (+ xn (car poi))
        y   (+ yn (cadr poi))
        z   (+ zn (caddr poi))
        pj  (list x y z)
        x1  (+ xxn (car pk1))
        y1  (+ yyn (cadr pk1))
        z1  (+ zzn (caddr pk1))
        pk  (list x1 y1 z1)
  )
  (repeat nel
    (3_CARA poi pj pk pk1)
    (setq poi pj
          pk1 pk
          x   (+ x xn)
          y   (+ y yn)
          z   (+ z zn)
          pj  (list x y z)
          x1  (+ x1 xxn)
          y1  (+ y1 yyn)
          z1  (+ z1 zzn)
          pk  (list x1 y1 z1)
    )
  )
)


;*****************************************************************************
;* * * LLENA UN CUADRILATERO P1 P2 P3 P4 CON 3DCARA
;*****************************************************************************

(defun omple4d (p1 p2 p3 p4 nl1   nl2
                /
                xn yn zn xd yd zd p11 p22 p33 p44 x y z)

  (setq xn  (/ (- (car p4) (car p1)) nl2)
        yn  (/ (- (cadr p4) (cadr p1)) nl2)
        zn  (/ (- (caddr p4) (caddr p1)) nl2)
        xd  (/ (- (car p3) (car p2)) nl2)
        yd  (/ (- (cadr p3) (cadr p2)) nl2)
        zd  (/ (- (caddr p3) (caddr p2)) nl2)
        p11 p1
        p22 p2
  )
  (repeat nl2
    (setq x   (+ xn (car p11))
          y   (+ yn (cadr p11))
          z   (+ zn (caddr p11))
          p44 (list x y z)
          x   (+ xd (car p22))
          y   (+ yd (cadr p22))
          z   (+ zd (caddr p22))
          p33 (list x y z)
    )
    (utira4 p11 p22 p33 p44 nl1)
    (setq p11 p44
          p22 p33
    )
  )
)
```

```
;*****************************************************************************
;* * *  DIBUJA ELEMENTOS 3Dcara DE CUATRO LADOS EN CUADRILATERO ALABEADO
;*****************************************************************************

(defun ph4d (/ p1 p2 p3 p4 nl1 nl2 dd)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq  p1 (getpoint "\nPrimera esquina:")
         p2 (getpoint p1 "\nSegunda esquina:")
         dd (distance p1 p2)
  )
  (princ "\n Longitud del lado 1º= ")
  (princ dd)
  (setq  nl1 (getint "\n Numero elementos en lado 1-2:")
         p3  (getpoint p2 "\nTercera  esquina:")
         dd  (distance p2 p3)
  )
  (princ "\n Longitud del lado 2º= ")
  (princ dd)
  (setq  nl2 (getint "\n Numero elementos en ese lado 2-3:")
         p4  (getpoint p3 "\nCuarta  esquina:")
  )
  (r_non)
  (omple4d p1 p2 p3 p4 nl1 nl2)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun C:ph4d () (ph4d))


;*****************************************************************************
;* * *  FUNCION DIBUJA UN MURO DE ALTURA H CON ELEM. FINITOS 4LADOS
;*****************************************************************************

(setq tph 0.4                           ;  tph   tamaño particion horizontal
      npv 8                             ;  npv   numero particiones vertical
      alt 3.0                           ;  alt   altura muro
)

(defun mur (/ pp p1 p2 p3 p4 d nh bb bbb f lm lz)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (scpu)
  (setq bbb npv)
  (princ "\n N Divisiones en vertical)<")
  (princ bbb)
  (princ "> ")
  (setq npv (getint))
  (if (eq (eval npv) nil)
    (setq npv bbb)
```

```
  )
  (setq bbb tph)
  (princ "\n N Tamaño Divisiones horizontal)<")
  (princ bbb)
  (princ "> ")
  (setq tph (getreal))
  (if (eq (eval tph) nil)
    (setq tph bbb)
  )
  (setq  p1 (getpoint "\n Primera esquina:")
         bb alt
  )
  (princ "\n N Altura en mt.)<")
  (princ bb)
  (princ "> ")
  (setq alt (getreal))
  (if (eq (eval alt) nil)
    (setq alt bb)
  )
  (setq  p4 (altp p1 alt)
         pp 1
  )
  (while (/= pp nil)
    (r_fmi)
    (setq p2 (getpoint p1 "\nSiguiente esquina:"))
    (setq pp p2)
    (if (/= pp nil)
      (progn (setq bb alt)
             (princ "\n N Altura en metros <")

             (princ bb)
             (princ "> ")
             (setq alt (getreal))
             (if (eq (eval alt) nil)
               (setq alt bb)
             )
             (setq p3 (altp p2 alt))
             (r_non)
             (setq d  (distance p1 p2)
                    nh (fix (/ d tph))
                    f  (omple4d p1 p2 p3 p4 nh npv)
                    lm (getvar "CLAYER")
                    lz (strcat lm "-zun")
             )
             (command "_LAYER" "_N" lz "")
             (command "_LAYER" "_S" lz "")
             (setq f (nnlin p3 p4 nh))
             (command "_LAYER" "_S" lm "")
      )
    )                                       ; de progn y de if
    (setq p1 p2
          p4 p3
    )
  )                                         ;de while

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:mur () (mur))


;*****************************************************************************
;* * *  RELLENA UNA ZONA DELIMITADA POR CUATRO ENTIDADES CON 3Dcara DE 4 LADOS
```

```
;*****************************************************************************

(defun m4 (/ s1 s2 v1 v2 v3 v4 bb)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)

  (setq  s1 (getvar "SURFTAB1")
         s2 (getvar "SURFTAB2")
  )
  (setq  v1 (entsel "\nSeleccione Primer Lado: ")
         bb s1
  )
  (princ "\n N Divisiones en direccion 1)<")
  (princ bb)
  (princ "> ")
  (setq s1 (getint))
  (if (eq (eval s1) nil)
    (setq s1 bb)
  )
  (setq  v2 (entsel "\nSeleccione Segundo Lado: ")
         bb s2
  )
  (princ "\n N Divisiones en direccion 2)<")
  (princ bb)
  (princ "> ")
  (setq s2 (getint))
  (if (eq (eval s2) nil)
    (setq s2 bb)
  )
  (setvar "SURFTAB1" s1)
  (setvar "SURFTAB2" s2)

  (setq  v3 (entsel "\nSeleccione Tercer Lado: ")
         v4 (entsel "\nSeleccione Cuarto Lado: ")
  )
  (command "_EDGESURF" v1 v2 v3 v4)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


(defun c:m4 () (m4))
```

6.2.6 Generación de Sistemas Estructurales por volúmenes

```
; *******   SISTEMAS ESTRUCTURALES DESARROLLADOS POR VOLUMENES

; pbase     Ver su definición para AutoCAD e IntelliCAD en Entorno.lsp
; tetra     Dibuja un tetraedro como malla poligonal dados los vértices
; s4        Pide los cuatro vértices de un tetraedro y lo dibuja como malla
;           poligonal
; c:s4
; s4a       DIBUJA UN SOLIDO TETRAEDRO DADOS LOS VERTICES DE LA BASE Y ALTURA
; c:s4a
```

```
; pr3       Rellena un prisma triangular con tetraedros
; c:pr3
; pr4       Rellena un prisma cuadrangular con tetraedros
; c:pr4
; s8        Pide los ocho vértices de un sólido de 6 caras y lo dibuja como
;           malla poligonal
; c:s8
; s8a       Pide los cuatro vértices de la base de un sólido de 6 caras y la
;           altura y lo dibuja como una malla poligonal
; c:s8a
; ffor      Funciones de forma para construir elemento serediptico de 20 nodos
; dibs20    Dibuja un hexaedro isoparamétrico
; utira8    Dibuja una tira de sólidos de 6 caras y 8 vértices
; utiran    Rellena una tira de sólidos de 6 caras y 8 vértices (coord. nat.)
; m8        Rellena un volumen dado por 8 vértices con elementos volumétricos
;           de tipo paralelepipédico
; c:m8
; m20       Rellena con sólidos un volumen definido por seis caras
;           de superficie curva
; c:m20     Idem a la función   m20
; m8nat     Prepara las coordenadas naturales
; corcar    Calcula las coord. cartesianas de un punto en coord. naturales



;*****************************************************************************
;* * *  DIBUJA UN TETRAEDRO COMO MALLA POLIGONAL DADOS LOS CUATRO VERTICES
;*****************************************************************************

(defun tetra ( p1 p2 p3 p4 / )

  (pbase p1 p2 p3 p3 p4 p4 p4 p4)

)


;*****************************************************************************
;* * *  DIBUJA UN SOLIDO TIPO TETRAEDRO DADOS LOS CUATRO VERTICES
;*****************************************************************************

(defun s4 (/ p1 p2 p3 p4 p5 p6 p7 p8)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq  p1 (getpoint "\n Primer vértice:")
         p2 (getpoint p1 "\n Segundo vértice:")
         p3 (getpoint p2 "\n Tercer  vértice:")
         p4 (getpoint p3 "\n Cuarto  vértice:")
  )
  (r_non)
  (pbase p1 p2 p3 p3 p4 p4 p4 p4)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:s4 () (s4))
```

```
;*****************************************************************************
;* * *  DIBUJA UN SOLIDO TETRAEDRO DADOS LOS VERTICES DE LA BASE Y LA ALTURA
;*****************************************************************************
(defun s4a (/ p1 p2 p3 p4 p5 p6 h1)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq  p1 (getpoint "\n Primer vértice:")
         p2 (getpoint p1 "\n Segundo vértice:")
         p3 (getpoint p2 "\n Tercer  vértice:")
         h1 (getreal "\n Altura :")
         p5 (pmig p2 p3)
         p6 (pmig p1 p5)
  )
  (r_non)
  (setq p4 (altp p6 h1))
  (pbase p1 p2 p3 p3 p4 p4 p4 p4)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:s4a () (s4a))


;*****************************************************************************
;* * *  DIBUJA SOLIDOs TETRAEDROs LLENANDO PRISMA TRIANGULAR
;*****************************************************************************
(defun pr3 (/ p1 p2 p3 p4 p5 p6)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq  p1 (getpoint "\n Primer vertice Triangulo 1:")
         p2 (getpoint p1 "\n Segundo vertice Triangulo 1:")
         p3 (getpoint p2 "\n Tercer vertice Triangulo 1:")
         p4 (getpoint p3 "\n Primer vertice Triangulo 2:")
         p5 (getpoint p4 "\n Segundo vertice Triangulo 2:")
         p6 (getpoint p5 "\n Tercer vertice Triangulo 2:")
  )
  (tetra p1 p2 p3 p5)
  (tetra p1 p4 p5 p3)
  (tetra p3 p6 p5 p4)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:pr3 () (pr3))
```

```
;*****************************************************************************
;* * *  FUNCION DIBUJA SOLIDOs TETRAEDROs LLENANDO PRISMA CUADRANGULAR
;*****************************************************************************

(defun pr4 (/ p1 p2 p3 p4 p5 p6 p7 p8 osm)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq  p1 (getpoint "\n Primer vertice Base de 4 vert:")
         p2 (getpoint p1 "\n Segundo vertice Base de 4 vert:")
         p3 (getpoint p2 "\n Tercer vertice Base de 4 vert:")
         p4 (getpoint p3 "\n Cuarto vertice Base de 4 vert:")
         p5 (getpoint p4 "\n Primer vertice Superior de 4 vert:")
         p6 (getpoint p5 "\n Segundo vertice Superior de 4 vert:")
         p7 (getpoint p6 "\n Tercer vertice Superior de 4 vert:")
         p8 (getpoint p7 "\n Cuarto vertice Superior de 4 vert:")
  )
  (tetra p1 p2 p4 p5)
  (tetra p2 p3 p4 p7)
  (tetra p2 p7 p6 p5)
  (tetra p4 p5 p7 p8)
  (tetra p2 p4 p5 p7)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:pr4 () (pr4))


;*****************************************************************************
;* * *  DIBUJA UN SOLIDO A PARTIR DE SUS OCHO VERTICES
;*****************************************************************************

(defun s8 (/ p1 p2 p3 p4 p5 p6 p7 p8 osm)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq  p1 (getpoint "\n Primer vértice:")
         p2 (getpoint p1 "\n Segundo vértice:")
         p3 (getpoint p2 "\n Tercer  vértice:")
         p4 (getpoint p3 "\n Cuarto  vértice:")
         p5 (getpoint p4 "\n Quinto vértice:")
         p6 (getpoint p5 "\n Sexto vértice:")
         p7 (getpoint p6 "\n Septimo vértice:")
         p8 (getpoint p7 "\n Octavo vértice:")
  )
  (r_non)
  (pbase p1 p2 p3 p4 p5 p6 p7 p8)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
```

```
)

(defun c:s8 () (s8))


;*****************************************************************************
;* * *  DIBUJA UN SOLIDO A PARTIR DE CUATRO VERTICES Y ALTURA
;*****************************************************************************

(defun s8a (/ p1 p2 p3 p4 p5 p6 p7 p8 h1 osm)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq  p1 (getpoint "\n Primer vértice:")
         p2 (getpoint p1 "\n Segundo vértice:")
         p3 (getpoint p2 "\n Tercer  vértice:")
         p4 (getpoint p3 "\n Cuarto  vértice:")
         h1 (getreal "\n Altura :")
  )
  (r_non)
  (setq  p5 (altp p1 h1)
         p6 (altp p2 h1)
         p7 (altp p3 h1)
         p8 (altp p4 h1)
  )
  (pbase p1 p2 p3 p4 p5 p6 p7 p8)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:s8a () (s8a))


;*****************************************************************************
;* * *  FUNCIONES DE FORMA ELEMENTO HEXAEDRICO SERENDIPTICO DE 20 NODOS
;*****************************************************************************

(defun ffor (s tt q / fform s2 t2 q2 xx yy ff)

  (setq  s2    (* s s)
         t2    (* tt tt)
         q2    (* q q)
         xx    (* (+ 1 s) (- 1 tt) (- 1 q) (- s tt q 2.0) 0.125)    ;N1
         yy    (* (- 1 t2) (+ 1 s) (- 1 q) 0.25)                    ;N2
         fform (list yy xx)
         xx    (* (+ 1 s) (+ 1 tt) (- 1 q) (- (+ s tt) q 2.0) 0.125)        ;N3
         fform (cons xx fform)
         yy    (* (- 1 s2) (+ 1 tt) (- 1 q) 0.25)                   ;N4
         fform (cons yy fform)
         xx    (* (- 1 s) (+ 1 tt) (- 1 q) (- tt s q 2.0) 0.125)    ;N5
         fform (cons xx fform)
         yy    (* (- 1 t2) (- 1 s) (- 1 q) 0.25)                    ;N6
         fform (cons yy fform)
         xx    (* (- 1 s) (- 1 tt) (- 1 q) (- 0.0 tt s q 2.0) 0.125)        ;N7
         fform (cons xx fform)
         yy    (* (- 1 s2) (- 1 tt) (- 1 q) 0.25)                   ;N8
         fform (cons yy fform)
         yy    (* (- 1 q2) (+ 1 s) (- 1 tt) 0.25)                   ;N9
```

```
        fform (cons yy fform)
        yy    (* (- 1 q2) (+ 1 s) (+ 1 tt) 0.25)                  ;N10
        fform (cons yy fform)
        yy    (* (- 1 q2) (- 1 s) (+ 1 tt) 0.25)                  ;N11
        fform (cons yy fform)
        yy    (* (- 1 q2) (- 1 s) (- 1 tt) 0.25)                  ;N12
        fform (cons yy fform)
        xx    (* (+ 1 s) (- 1 tt) (+ 1 q) (- (+ s q) tt 2.0) 0.125);N13
        fform (cons xx fform)
        yy    (* (- 1 t2) (+ 1 s) (+ 1 q) 0.25)                   ;N14
        fform (cons yy fform)
        xx    (* (+ 1 s) (+ 1 tt) (+ 1 q) (- (+ s tt q) 2.0) 0.125);N15
        fform (cons xx fform)
        yy    (* (- 1 s2) (+ 1 tt) (+ 1 q) 0.25)                  ;N16
        fform (cons yy fform)
        xx    (* (- 1 s) (+ 1 tt) (+ 1 q) (- (+ tt q) s 2.0) 0.125);N17
        fform (cons xx fform)
        yy    (* (- 1 t2) (- 1 s) (+ 1 q) 0.25)                   ;N18
        fform (cons yy fform)
        xx    (* (- 1 s) (- 1 tt) (+ 1 q) (- q tt s 2.0) 0.125)   ;N19
        fform (cons xx fform)
        yy    (* (- 1 s2) (- 1 tt) (+ 1 q) 0.25)                  ;N20
        fform (cons yy fform)
        ff    (reverse fform)
  )
)


;*************************************************************************
;* * *  FUNCION QUE DIBUJA UN HEXAEDRO ISOPARAMETRICO
;*************************************************************************

(defun dibs20 (cor20   p1  p2  p3  p4  p5  p6  p7  p8  /   xn  yn  zn
               px  py  pz  x   y   z   d1  d2  d3  d4  d5  d6  d7  d8
               ffm
              )

  (setq xn  (nth 0 p1)
        yn  (nth 1 p1)
        zn  (nth 2 p1)
        ffm (ffor xn yn zn)
        d1  (corcar cor20 ffm)
        xn  (nth 0 p2)
        yn  (nth 1 p2)
        zn  (nth 2 p2)
        ffm (ffor xn yn zn)
        d2  (corcar cor20 ffm)
        xn  (nth 0 p3)
        yn  (nth 1 p3)
        zn  (nth 2 p3)
        ffm (ffor xn yn zn)
        d3  (corcar cor20 ffm)
        xn  (nth 0 p4)
        yn  (nth 1 p4)
        zn  (nth 2 p4)
        ffm (ffor xn yn zn)
        d4  (corcar cor20 ffm)
        xn  (nth 0 p5)
        yn  (nth 1 p5)
        zn  (nth 2 p5)
        ffm (ffor xn yn zn)
        d5  (corcar cor20 ffm)
        xn  (nth 0 p6)
        yn  (nth 1 p6)
        zn  (nth 2 p6)
        ffm (ffor xn yn zn)
```

```
        d6  (corcar cor20 ffm)
        xn  (nth 0 p7)
        yn  (nth 1 p7)
        zn  (nth 2 p7)
        ffm (ffor xn yn zn)
        d7  (corcar cor20 ffm)
        xn  (nth 0 p8)
        yn  (nth 1 p8)
        zn  (nth 2 p8)
        ffm (ffor xn yn zn)
        d8  (corcar cor20 ffm)
  )
  (pbase d1 d2 d3 d4 d5 d6 d7 d8)
)


;*****************************************************************************
;* * * FUNCION PARA DIBUJAR UNA TIRA DE SOLIDOS 6 CARAS 8 VERTICES
;*****************************************************************************

(defun utira8 (p1   p2    p3   p4   p5    p6   p7   p8   nel  /     xn
               yn   zn    poi  pj   pk    pl   x    y    z    xxn   yyn
               zzn  xs    ys   zs   xs1   ys1  zs1  x1   y1   z1    xn1
               yn1  zn1   xxn1 yyn1 zzn1  pii  pjj  pkk  pll
              )

  (setq  xn  (/ (- (car p2) (car p1)) nel)
         yn  (/ (- (cadr p2) (cadr p1)) nel)
         zn  (/ (- (caddr p2) (caddr p1)) nel)
         xxn (/ (- (car p3) (car p4)) nel)
         yyn (/ (- (cadr p3) (cadr p4)) nel)
         zzn (/ (- (caddr p3) (caddr p4)) nel)
         poi p1
         pl  p4
         x   (+ xn (car poi))
         y   (+ yn (cadr poi))
         z   (+ zn (caddr poi))
         pj  (list x y z)
         x1  (+ xxn (car pl))
         y1  (+ yyn (cadr pl))
         z1  (+ zzn (caddr pl))
         pk  (list x1 y1 z1)
  )
  (setq  xn1  (/ (- (car p6) (car p5)) nel)
         yn1  (/ (- (cadr p6) (cadr p5)) nel)
         zn1  (/ (- (caddr p6) (caddr p5)) nel)
         xxn1 (/ (- (car p7) (car p8)) nel)
         yyn1 (/ (- (cadr p7) (cadr p8)) nel)
         zzn1 (/ (- (caddr p7) (caddr p8)) nel)
         pii  p5
         pll  p8
         xs   (+ xn1 (car pii))
         ys   (+ yn1 (cadr pii))
         zs   (+ zn1 (caddr pii))
         pjj  (list xs ys zs)
         xs1  (+ xxn1 (car pll))
         ys1  (+ yyn1 (cadr pll))
         zs1  (+ zzn1 (caddr pll))
         pkk  (list xs1 ys1 zs1)
  )
  (repeat nel
    (pbase poi pj pk pl pii pjj pkk pll)
    (setq poi pj
          pl  pk
          x   (+ x xn)
          y   (+ y yn)
```

```
          z   (+ z zn)
          pj  (list x y z)
          x1  (+ x1 xxn)
          y1  (+ y1 yyn)
          z1  (+ z1 zzn)
          pk  (list x1 y1 z1)
    )
    (setq pii pjj
          pll pkk
          xs  (+ xs xn1)
          ys  (+ ys yn1)
          zs  (+ zs zn1)
          pjj (list xs ys zs)
          xs1 (+ xs1 xxn1)
          ys1 (+ ys1 yyn1)
          zs1 (+ zs1 zzn1)
          pkk (list xs1 ys1 zs1)
    )
  )
)


;*************************************************************************
;* * * * * FUNCION UNA TIRA DE SOLIDOS 6 CARAS 8 VERTICES cord naturales
;*************************************************************************

(defun utiran (cor20      p1   p2   p3    p4   p5    p6   p7   p8    nel
               /    xn    yn   zn   ppi   pj   pk    pl   x    y     z
               xxn  yyn   zzn  xs   ys    zs   xs1   ys1  zs1  x1    y1
               z1   xn1   yn1  zn1  xxn1  yyn1 zzn1 pii  pjj  pkk   pll
              )

  (setq xn  (/ (- (car p2) (car p1)) nel)
        yn  (/ (- (cadr p2) (cadr p1)) nel)
        zn  (/ (- (caddr p2) (caddr p1)) nel)
        xxn (/ (- (car p3) (car p4)) nel)
        yyn (/ (- (cadr p3) (cadr p4)) nel)
        zzn (/ (- (caddr p3) (caddr p4)) nel)
        ppi p1
        pl  p4
        x   (+ xn (car ppi))
        y   (+ yn (cadr ppi))
        z   (+ zn (caddr ppi))
        pj  (list x y z)
        x1  (+ xxn (car pl))
        y1  (+ yyn (cadr pl))
        z1  (+ zzn (caddr pl))
        pk  (list x1 y1 z1)
  )                                    ;--------------------------

  (setq xn1  (/ (- (car p6) (car p5)) nel)
        yn1  (/ (- (cadr p6) (cadr p5)) nel)
        zn1  (/ (- (caddr p6) (caddr p5)) nel)
        xxn1 (/ (- (car p7) (car p8)) nel)
        yyn1 (/ (- (cadr p7) (cadr p8)) nel)
        zzn1 (/ (- (caddr p7) (caddr p8)) nel)
        pii  p5
        pll  p8
        xs   (+ xn1 (car pii))
        ys   (+ yn1 (cadr pii))
        zs   (+ zn1 (caddr pii))
        pjj  (list xs ys zs)
        xs1  (+ xxn1 (car pll))
        ys1  (+ yyn1 (cadr pll))
        zs1  (+ zzn1 (caddr pll))
        pkk  (list xs1 ys1 zs1)
```

```
  )                                     ;-------------------------

  (repeat nel
    (dibs20 cor20 ppi pj pk pl pii pjj pkk pll)
    (setq ppi pj
          pl  pk
          x   (+ x xn)
          y   (+ y yn)
          z   (+ z zn)
          pj  (list x y z)
          x1  (+ x1 xxn)
          y1  (+ y1 yyn)
          z1  (+ z1 zzn)
          pk  (list x1 y1 z1)
    )                                   ;-------------------------

    (setq pii pjj
          pll pkk
          xs  (+ xs xn1)
          ys  (+ ys yn1)
          zs  (+ zs zn1)
          pjj (list xs ys zs)
          xs1 (+ xs1 xxn1)
          ys1 (+ ys1 yyn1)
          zs1 (+ zs1 zzn1)
          pkk (list xs1 ys1 zs1)
    )
  )
)


;*****************************************************************************
;* * *  FUNCION PARA LLENAR UN VOLUMEN CON SOLIDOS
;*****************************************************************************

(defun m8 (/     p1   p2   p3   p4   p5    p6   p7   p8    nh   nt   nl1
           nl2   nl3  pb1  pb2  pb3  pb4   ps1  ps2  ps3   ps4  pbf3 pbf4
           psf3  psf4 pss1 pss2 osm
          )

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_fmi)

  (setq  p1  (getpoint "\nBASE Primera esquina:")
         p2  (getpoint p1 "\nBASE Segunda esquina:")
         nl1 (getint "\nNumero elementos en lado 1-2:")
         p3  (getpoint p2 "\nBASE Tercera  esquina:")
         nl2 (getint "\nNumero elementos en lado 2-3:")
         p4  (getpoint p3 "\nBASE Cuarta  esquina:")
         p5  (getpoint p4 "\nSUPERIOR Punto 1:")
         p6  (getpoint p5 "\nSUPERIOR Punto 2:")
         p7  (getpoint p6 "\nSUPERIOR Punto 3:")
         p8  (getpoint p7 "\nSUPERIOR Punto 4:")
         nl3 (getint "\n Numero de elementos en altura: ")
  )
  (r_non)
  (setq  nh   (+ nl3 1)
         pb1  p1
         pb2  p2
         pbf3 p3
         pbf4 p4
  )
```

```
  (repeat nl3
    (setq nh   (- nh 1)
          pss1 (pfrac pb1 p5 nh)
          pss2 (pfrac pb2 p6 nh)
          psf3 (pfrac pbf3 p7 nh)
          psf4 (pfrac pbf4 p8 nh)
          nt   (+ nl2 1)
          ps1  pss1
          ps2  pss2
    )
    (repeat nl2
      (setq nt   (- nt 1)
            pb3  (pfrac pb2 pbf3 nt)
            pb4  (pfrac pb1 pbf4 nt)
            ps3  (pfrac ps2 psf3 nt)
            ps4  (pfrac ps1 psf4 nt)
      )
      (utira8 pb1 pb2 pb3 pb4 ps1 ps2 ps3 ps4 nl1)
      (setq pb1  pb4
            pb2  pb3
            ps1  ps4
            ps2  ps3
      )
    )                                       ; final de repeat nl2
    (setq pbf4 psf4
          pbf3 psf3
          pb1  pss1
          pb2  pss2
    )
  )
                                            ; final de repeat nl3
  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:m8 () (m8))

; m20    RELLENA UN VOLUMEN DE CARAS CURVAS CON SOLIDOS

;*****************************************************************************
;* * *  Rellena con sólidos volumen definido por seis caras de superficie curva
;*****************************************************************************

(defun m20 (/     ent  lp   p1    p2   p3   p4   p5   p6    p7   p8   p9
            p10  p11  p12  p13   p14  p15  p16  p17  p18   p19  p20  c20
            pp1  pp2  pp3  nl1   nl2  nl3  om
           )
  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)

  (scpu)
  (setq ent (entsel "\nPRIMER lado de la base : ")
        lp  (l3p ent)
  )
  (scpu)
  (setq p11 (nth 0 lp)
        p13 (nth 1 lp)
        p2  (nth 2 lp)
        nl1 (getint "\nNumero elementos en lado 1-2:")
        ent (entsel "\nSEGUNDO lado de la base : ")
```

```
      lp  (l3p ent)
      pp1 (nth 0 lp)
      pp2 (nth 1 lp)
      p4  (nth 2 lp)
)
(if (or (< (distance p13 pp1) 0.03) (< (distance p13 pp2) 0.03))
  (setq p1 p11
        p3 p13
  )
  (setq p1 p13
        p3 p11
  )
)
(if (< (distance p3 pp1) 0.03)
  (setq p5 (nth 1 lp))
  (setq p5 (nth 0 lp))
)
(grdraw p2 p3 1 1)
(setq nl2 (getint "\nNumero elementos en lado 2-3:"))
(grdraw p4 p5 1 1)
(setq  ent (entsel "\nTERCER lado de la base : ")
      lp  (l3p ent)
      pp1 (nth 0 lp)
      pp2 (nth 1 lp)
      p6  (nth 2 lp)
)
(if (< (distance p5 pp1) 0.03)
  (setq p7 (nth 1 lp))
  (setq p7 (nth 0 lp))
)
(grdraw p6 p7 1 1)
(setq  ent (entsel "\nCUARTO lado de la base : ")
      lp  (l3p ent)
      p8  (nth 2 lp)
)
(grdraw p1 p2 2 2)
(setq  ent (entsel "\nLADO VERTICAL 1ºVERTICE : ")
      lp  (l3p ent)
      pp1 (nth 0 lp)
      pp2 (nth 1 lp)
      p9  (nth 2 lp)
)
(if (< (distance p1 pp1) 0.03)
  (setq p13 (nth 1 lp))
  (setq p13 (nth 0 lp))
)
(grdraw p9 p13 1 1)
(setq nl3 (getint "\nNumero elementos en lado 1-5(VERTICAL):"))
(grdraw p3 p4 2 2)
(setq ent (entsel "\nLADO VERTICAL 2ºVERTICE : "))
(setq lp (l3p ent))
(setq  pp1 (nth 0 lp)
      pp2 (nth 1 lp)
      p10 (nth 2 lp)
)
(if (< (distance p3 pp1) 0.03)
  (setq p15 (nth 1 lp))
  (setq p15 (nth 0 lp))
)
(grdraw p10 p15 1 1)
(grdraw p5 p6 2 2)
(setq  ent (entsel "\nLADO VERTICAL 3ºVERTICE : ")
      lp  (l3p ent)
      pp1 (nth 0 lp)
      pp2 (nth 1 lp)
      p11 (nth 2 lp)
```

```
  )
  (if (< (distance p5 pp1) 0.03)
    (setq p17 (nth 1 lp))
    (setq p17 (nth 0 lp))
  )
  (grdraw p11 p17 1 1)
  (grdraw p7 p8 2 2)
  (setq ent (entsel "\nLADO VERTICAL 4ºVERTICE : ")
        lp  (l3p ent)
        pp1 (nth 0 lp)
        pp2 (nth 1 lp)
        p12 (nth 2 lp)
  )
  (if (< (distance p7 pp1) 0.03)
    (setq p19 (nth 1 lp))
    (setq p19 (nth 0 lp))
  )
  (grdraw p9 p13 1 1)
  (setq ent (entsel "\nCARA SUPERIOR Lado 1-2 : ")
        lp  (l3p ent)
        p14 (nth 2 lp)
  )
  (grdraw p13 p14 2 2)
  (setq ent (entsel "\nCARA SUPERIOR Lado 2-3 : "))
  (setq lp  (l3p ent)
        p16 (nth 2 lp)
  )
  (grdraw p15 p16 2 2)
  (setq ent (entsel "\nCARA SUPERIOR Lado 3-4 : ")
        lp  (l3p ent)
        p18 (nth 2 lp)
  )
  (grdraw p17 p18 2 2)
  (setq ent (entsel "\nCARA SUPERIOR Lado 4-1 : ")
        lp  (l3p ent)
        p20 (nth 2 lp)
  )
  (grdraw p19 p20 2 2)
  (scpu)
  (setq c20 (list p1   p2   p3    p4   p5   p6    p7   p8   p9   p10
                  p11  p12  p13   p14  p15  p16   p17  p18  p19  p20
                 )
  )
  (m8nat c20 nl2 nl1 nl3)

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

(defun c:m20 () (m20))


;*****************************************************************************
;* * *  PREPARA LAS COORDENADAS NATURALES
;*****************************************************************************

(defun m8nat (cor20      nl1  nl2  nl3  /    p1    p2   p3   p4    p5
              p6   p7    p8   nh    nt   pb1  pb2   pb3  pb4  ps1   ps2
              ps3  ps4   pbf3 pbf4 psf3 psf4 pss1 pss2 om
             )

  (cprev)
  (noecho)
  (cposm)
```

```
  (cpscp)
  (cpcap)
  (r_non)

  (setq  p1   (list -1.0 -1.0 -1.0)
         p2   (list 1.0 -1.0 -1.0)
         p3   (list 1.0 1.0 -1.0)
         p4   (list -1.0 1.0 -1.0)
         p5   (list -1.0 -1.0 1.0)
         p6   (list 1.0 -1.0 1.0)
         p7   (list 1.0 1.0 1.0)
         p8   (list -1.0 1.0 1.0)
         nh   (+ nl3 1)
         pb1  p1
         pb2  p2
         pbf3 p3
         pbf4 p4
  )
  (repeat nl3
    (setq nh   (- nh 1)
          pss1 (pfrac pb1 p5 nh)
          pss2 (pfrac pb2 p6 nh)
          psf3 (pfrac pbf3 p7 nh)
          psf4 (pfrac pbf4 p8 nh)
          nt   (+ nl2 1)
          ps1  pss1
          ps2  pss2
    )
    (repeat nl2
      (setq nt   (- nt 1)
            pb3  (pfrac pb2 pbf3 nt)
            pb4  (pfrac pb1 pbf4 nt)
            ps3  (pfrac ps2 psf3 nt)
            ps4  (pfrac ps1 psf4 nt)
      )
      (utiran cor20 pb1 pb2 pb3 pb4 ps1 ps2 ps3 ps4 nl1)
      (setq pb1  pb4
            pb2  pb3
            ps1  ps4
            ps2  ps3
      )
    )                                       ; final de repeat nl2
    (setq pbf4 psf4
          pbf3 psf3
          pb1  pss1
          pb2  pss2
    )
  )                                         ; final de repeat nl3

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


;*****************************************************************************
;* * * CALCULA LAS COORDENADAS CARTESIANAS DE UN PUNTO EN COORD. NATURALES
;*****************************************************************************

(defun corcar (cor20 ffm / x y z xi yi zi pc cont ni cpp)

  (setq  x 0.0
         y 0.0
         z 0.0
         cont 0
```

```
  )
  (repeat 20
    (setq pc   (nth cont cor20)
          ni   (nth cont ffm)
          xi   (nth 0 pc)
          yi   (nth 1 pc)
          zi   (nth 2 pc)
          x    (+ x (* xi ni))
          y    (+ y (* yi ni))
          z    (+ z (* zi ni))
          cont (1+ cont)
    )
  )
  (setq cpp (list x y z))
)
```

6.2.7 Generación de Vigas y Porticos

```
; ************* VIGAS Y PORTICOS

; vanos         Solicita las luces correspondientes a un número V de vanos
; plants        Solicita las alturas correspondientes a un número H de plantas
; vigacont      Dibuja una viga continua con V vanos con o sin dimensiones
; portico       Pórtico plano con V vanos y H plantas con o sin dimensiones
; portico3      Pórtico en 3D con X vanos H plantas e Y planos
;               con o sin dimensiones


;*****************************************************************************
;* * * SOLICITA LAS LUCES CORRESPONDIENTES A UN NUMERO v DE VANOS
;*****************************************************************************

(defun vanos (v / dv s i f)

  (if (= v 0)
    (progn (setq dv (list 0.0)
                 f  (list 0 0 0)
           )
           (setq i (getpoint "\nExtremo INICIAL del primer vano"))
           (while (/= nil f)
             (setq f (getpoint "\nExtremo FINAL del vano"))
             (if (/= nil f)
               (setq dv (cons (distance i f) dv))
             )
             (setq i f)
           )
           (print (cdr (reverse dv)))
           (cdr (reverse dv))
    )
    (progn (prompt
             "\nLuz de cada vano en metros. Comience por la izquierda. "
           )
           (prompt "( Ejemplo   4.50 4.50 4.50 )")
           (prompt
             "\nSi es la misma anote la luz precedida del signo  *=  ."
           )
           (prompt " ( Ejemplo   *=4.50  )")
```

```
          (setq dv (getstring T "\nLuces:    "))
          (if (= "" dv)
            (setq dv "*=1.00")
          )
          (if (= "*=" (substr dv 1 2))
            (progn (setq dv (substr dv 3)
                         s  (strcat dv " ")
                   )
                   (repeat (- v 1) (setq dv (strcat s dv)))
            )
          )
          (setq s (strcat "(list " dv ")"))
          (setq dv (eval (read s)))
    )
  )
)


;*****************************************************************************
;* * * SOLICITA LAS ALTURAS CORRESPONDIENTES A UN NUMERO h DE PLANTAS
;*****************************************************************************

(defun plants (h / dh s i f)

  (if (= h 0)
    (progn (setq dh (list 0.0)
                  f  (list 0 0 0)
           )
           (setq i (getpoint "\nExtremo INFERIOR de la primera planta"))
           (while (/= nil f)
             (setq f (getpoint "\nExtremo FINAL de la planta"))
             (if (/= nil f)
               (setq dh (cons (distance i f) dh))
             )
             (setq i f)
           )
           (print (cdr (reverse dh)))
           (cdr (reverse dh))
    )
    (progn
      (prompt
        "\nAltura de cada planta en metros. Comience por la planta baja."
      )
      (prompt "( Ejemplo   3.00 3.00 3.00 )")
      (prompt
        "\nSi es la misma anote la altura precedida del signo   *=    ."
      )
      (prompt " ( Ejemplo   *=4.50 )")
      (setq dh (getstring T "\nAlturas:   "))
      (if (= "" dh)
        (setq dh "*=1.00")
      )
      (if (= "*=" (substr dh 1 2))
        (progn (setq dh  (substr dh 3)
                     s   (strcat dh " ")
               )
               (repeat (- h 1) (setq dh (strcat s dh)))
        )
      )
      (setq s (strcat "(list " dh ")"))
      (setq dh (eval (read s)))
    )
  )
)
```

```
;*****************************************************************************
;* * * DIBUJA UNA VIGA CONTINUA CON v VANOS CON O SIN DIMENSIONES
;*****************************************************************************

(defun vigacont  (/ v dv o p q)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)

  (setq  o (getpoint
             "\nPunto de inserción de un extremo de la viga continua "
           )
  )

  (while (not (or (> v 0) (= nil v)))
    (setq v (getint "\nNúmero de vanos "))
  )
  (if (= nil v)
    (progn (setq dv (vanos 0)
                 v  (length dv)
           )
    )
    (progn (setq dv (vanos v))
           (while (/= v (length dv))
             (prompt "\nEl número de vanos no coincide ")
             (prompt "con el número de luces ")
             (setq dv (vanos v))
           )
    )
  )

  (if (and (/= nil v) (/= nil dv))
    (progn (setq p o)
           (repeat v
             (setq q (mapcar '+ (list (car dv) 0.0 0.0) p))
             (command "_LINE" p q "")
             (setq p  q
                   dv (cdr dv)
             )
           )
    )
  )
  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


;*****************************************************************************
;* * * DIBUJA UN PORTICO PLANO CON v VANOS Y h PLANTAS CON O SIN DIMENSIONES
;*****************************************************************************

(defun portico (/ v h dv dh i j l o p q)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)
  (>?)
```

```
(setq  v -1
       h -1
)
(setq o (getpoint "\nPunto de inserci¢n del pórtico "))

(while (not (or (> v 0) (= nil v)))
  (setq v (getint "\nNúmero de vanos "))
)
(if (= nil v)
  (progn (setq dv (vanos 0)
                 v  (length dv)
         )
  )
  (progn (setq dv (vanos v))
         (while (/= v (length dv))
           (prompt "\nEl número de vanos no coincide ")
           (prompt "con el número de luces ")
           (setq dv (vanos v))
         )
  )
)

(while (not (or (> h 0) (= nil h)))
  (setq h (getint "\nNúmero de plantas "))
)
(if (= nil h)
  (progn (setq dh (plants 0)
                 h  (length dh)
         )
  )
  (progn (setq dh (plants h))
         (while (/= h (length dh))
           (prompt "\nEl número de plantas no coincide ")
           (prompt "con el número de alturas ")
           (setq dh (plants h))
         )
  )
)

(if (and (/= nil v) (/= nil dv) (/= nil h) (/= nil dh))
  (progn (setq i dv
               p o
         )
         (if (= (tblsearch "LAYER" "STR01") nil)
           (command "_LAYER" "_N" "STR01" "_COLOR" "2" "STR01" "")
         )
         (if (= (tblsearch "LAYER" "STR02") nil)
           (command "_LAYER" "_N" "STR02" "_COLOR" "3" "STR02" "")
         )
         (repeat v
           (setq j dh)
           (repeat h
             (setq q (mapcar '+ (list 0.0 (car j) 0.0) p)
                   l (mapcar '+ (list (car i) 0.0 0.0) q)
             )
             (command  "_LAYER"  "_T"       "STR01"   "_ON"
                       "STR01"    "_S"       "STR01"   ""
                     )
             (command "_LINE" p q "")
             (command  "_LAYER"  "_T"       "STR02"   "_ON"
                       "STR02"    "_S"       "STR02"   ""
                     )
             (command "_LINE" q l "")
             (setq j (cdr j)
                   p q
             )
```

```
            )
            (setq p (mapcar '+ (list (car i) 0.0 0.0) o)
                  o p
            )
            (setq i (cdr i))
          )

          (setq j dh)
          (repeat h
            (setq q (mapcar '+ (list 0.0 (car j) 0.0) p))
            (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01"
                     "")
            (command "_LINE" p q "")
            (setq j (cdr j)
                  p q
            )
          )
    )
  )


  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


;*****************************************************************************
;* * * DIBUJA PORTICO 3D CON x VANOS h PLANTAS E y PLANOS CON O SIN DIMENSIONES
;*****************************************************************************

(defun portico3  (/ x y h dx dy dh i j k l m o p q)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)
  (>?)

  (setq  x -1
         y -1
         h -1
  )
  (setq o (getpoint "\nPunto de inserción del pórtico "))

  (while (not (or (> x 0) (= nil x)))
    (setq x (getint "\nNúmero de vanos en dirección X"))
  )
  (if (= nil x)
    (progn (setq dx (vanos 0)
                 x  (length dx)
           )
    )
    (progn (setq dx (vanos x))
           (while (/= x (length dx))
             (prompt "\nEl número de vanos no coincide ")
             (prompt "con el número de luces ")
             (setq dx (vanos x))
           )
    )
  )

  (while (not (or (> y 0) (= nil y)))
```

```
  (setq y (getint "\nNúmero de vanos en dirección Y"))
)
(if (= nil y)
  (progn (setq dy (vanos 0)
               y  (length dy)
         )
  )
  (progn (setq dy (vanos y))
         (while (/= y (length dy))
           (prompt "\nEl número de vanos no coincide ")
           (prompt "con el número de luces ")
           (setq dy (vanos y))
         )
  )
)

(while (not (or (> h 0) (= nil h)))
  (setq h (getint "\nNúmero de plantas "))
)
(command "_UCS" "x" "90")
(if (= nil h)
  (progn (setq dh (plants 0)
               h  (length dh)
         )
  )
  (progn (setq dh (plants h))
         (while (/= h (length dh))
           (prompt "\nEl número de plantas no coincide ")
           (prompt "con el número de alturas ")
           (setq dh (plants h))
         )
  )
)
(scpu)

(if (and (/= nil x)
         (/= nil dx)
         (/= nil y)
         (/= nil dy)
         (/= nil h)
         (/= nil dh)
    )
  (progn (setq i dx
               j dy
               p o
               n o
               g o
         )
         (if (= (tblsearch "LAYER" "STR01") nil)
           (command "_LAYER" "_N" "STR01" "_COLOR" "2" "STR01" "")
         )
         (if (= (tblsearch "LAYER" "STR02") nil)
           (command "_LAYER" "_N" "STR02" "_COLOR" "3" "STR02" "")
         )
         (if (= (tblsearch "LAYER" "STR03") nil)
           (command "_LAYER" "_N" "STR03" "_COLOR" "7" "STR03" "")
         )
         (repeat y
           (repeat x
             (setq k dh)
             (repeat h
               (setq q (mapcar '+ (list 0.0 0.0 (car k)) p)
                     l (mapcar '+ (list (car i) 0.0 0.0) q)
                     m (mapcar '+ (list 0.0 (car j) 0.0) q)
               )
               (command "_LAYER"  "_T"      "STR01"     "_ON"
```

```
                "STR01"   "_S"      "STR01"     ""
               )
        (command "_LINE" p q "")
        (command "_LAYER"  "_T"      "STR02"     "_ON"
                 "STR02"   "_S"      "STR02"     ""
                )
        (command "_LINE" q l "")
        (command "_LAYER"  "_T"      "STR03"     "_ON"
                 "STR03"   "_S"      "STR03"     ""
                )
        (command "_LINE" q m "")
        (setq k (cdr k)
              p q
        )
      )
      (setq p (mapcar '+ (list (car i) 0.0 0.0) n)
            n p
      )
      (setq i (cdr i))
    )
    (setq k dh)
    (repeat h
      (setq q (mapcar '+ (list 0.0 0.0 (car k)) p)
            m (mapcar '+ (list 0.0 (car j) 0.0) q)
      )
      (command  "_LAYER"  "_T"       "STR01"   "_ON"
                "STR01"   "_S"       "STR01"   ""
               )
      (command "_LINE" p q "")
      (command  "_LAYER"  "_T"       "STR03"   "_ON"
                "STR03"   "_S"       "STR03"   ""
               )
      (command "_LINE" q m "")
      (setq k (cdr k)
            p q
      )
    )
    (setq i dx)
    (setq n (mapcar '+ (list 0.0 (car j) 0.0) o)
          o n
    )
    (setq p (mapcar '+ (list 0.0 (car j) 0.0) ¤)
          ¤ p
    )
    (setq j (cdr j))
  )
  (setq o p)
  (repeat x
    (setq k dh)
    (repeat h
      (setq q (mapcar '+ (list 0.0 0.0 (car k)) p)
            l (mapcar '+ (list (car i) 0.0 0.0) q)
      )
      (command  "_LAYER"  "_T"       "STR01"   "_ON"
                "STR01"   "_S"       "STR01"   ""
               )
      (command "_LINE" p q "")
      (command  "_LAYER"  "_T"       "STR02"   "_ON"
                "STR02"   "_S"       "STR02"   ""
               )
      (command "_LINE" q l "")
      (setq k (cdr k)
            p q
      )
    )
    (setq p (mapcar '+ (list (car i) 0.0 0.0) o)
```

```
                    o p
                    i (cdr i)
              )
            )
            (setq k dh)
            (repeat h
              (setq q (mapcar '+ (list 0.0 0.0 (car k)) p))
              (command "_LAYER" "_T" "STR01" "_ON" "STR01" "_S" "STR01"
                       "")
              (command "_LINE" p q "")
              (setq k (cdr k)
                    p q
              )
            )
    )
  )

  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)
```

6.2.8 Generación de Forjados Reticulares

```
; ************* GENERACION DE FORJADOS RETICULARES


; c:RS       GENERA UN RECUADRO PARA FORJADO RETICULAR CON ABACOS
; c:ABACOC   ABACO CENTRAL
; c:ABACOB   ABACO DE BORDE
; c:ABACOE   ABACO DE BORDE
; abac       GENERA LOS 3DACARA DE UN ABACO BASE EN FORJ RETICULAR
; c:ZUNB     GENERA UN ZUNCHO EN BORDE DE RECUADRO RETICULAR CON ABACOS


;*****************************************************************************
;* * *   GENERA UN RECUADRO PARA FORJADO RETICULAR CON ABACOS
;*****************************************************************************

(defun c:RS (/     p1   p2   p3    p4   lrr  lab   p11  p12  p21  p22
             p31  p32  p41  p42   px   py   pxx   pyy  fr   f1r  bb
             pp
            )

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)

  (r_fmi)
  (scpu)

  (setq  fr 0.15
         p1 (getpoint "\nRECUADRO    Primera esquina:")
         pp p1
  )
  (while (/= pp nil)
    (if (/= pp nil)
      (progn
        (setq p2 (getpoint p1 "\n            Segunda esquina:")
```

```
            p3 (getpoint p2 "\n           Tercera  esquina:")
            p4 (getpoint p3 "\n            Cuarta  esquina:")
            bb fr
      )
      (princ "\n Abaco fraccion de la luz <")
      (princ bb)
      (princ "> ")
      (setq fr (getreal))
      (if (eq (eval fr) nil)
        (setq fr bb)
      )
      (setq f1r (- 1 fr))
      (r_non)
      (setq l   (getvar "CLAYER")
            lrr (strcat l "-rec")
            lab (strcat l "-abac")
      )
      (command "_LAYER" "_N" lab "")
      (command "_LAYER" "_N" lrr "")
      (setq p11 (puntint p1 p2 fr)
            p12 (puntint p1 p2 f1r)
            p21 (puntint p2 p3 fr)
            p22 (puntint p2 p3 f1r)
            p31 (puntint p3 p4 fr)
            p32 (puntint p3 p4 f1r)
            p41 (puntint p4 p1 fr)
            p42 (puntint p4 p1 f1r)
            px  (puntint p21 p42 fr)
            py  (puntint p22 p41 fr)
            pxx (puntint p11 p32 fr)
            pyy (puntint p11 p32 f1r)
      )
      (command "_LAYER" "_S" lrr "")
      (omple4d pxx px py pyy 6 6)
      (omple4d pxx p11 p12 px 2 6)
      (omple4d px p21 p22 py 2 6)
      (omple4d py p31 p32 pyy 2 6)
      (omple4d p41 pyy pxx p42 2 6)
      (command "_LAYER" "_S" lab "")
      (abac p1 p11 pxx p42)
      (abac p2 p21 px p12)
      (abac p3 p31 py p22)
      (abac p4 p41 pyy p32)
      (command "_LAYER" "_S" l "")
    )
  )
  (r_fmi)
  (setq p1 (getpoint "\nRECUADRO    Primera esquina:")
        pp p1
  )
 )
 (avisoUNDO)
 (pgosm)
 (pgscp)
 (pgcap)
)

;*****************************************************************************
;* * *   ABACO CENTRAL
;*****************************************************************************

(defun c:ABACOC  (/ p1 p2 p3 p4 lab p11 p12 p23 p34 p41 pp)

  (cprev)
  (noecho)
```

```
  (cposm)
  (cpscp)
  (cpcap)

  (scpu)
  (r_fmi)

  (setq  p1 (getpoint "\nABACO CENTRAL Primera esquina :")
         pp p1
  )
  (while (/= pp nil)
    (if  (/= pp nil)
      (progn
        (setq p2  (getpoint p1 "\n              Segunda esquina :")
              p3  (getpoint p2 "\n              Tercera  esquina:")
              p4  (getpoint p3 "\n              Cuarta  esquina :")
              p11 (pmig p1 p3)
              p12 (pmig p1 p2)
              p23 (pmig p2 p3)
              p34 (pmig p3 p4)
              p41 (pmig p4 p1)
        )
        (r_non)
        (setq l   (getvar "CLAYER")
              lrr (strcat l "-rec")
              lab (strcat l "-abac")
        )
        (command "_LAYER" "_N" lab "_S" lab "")
        (abac p11 p41 p1 p12)
        (abac p11 p12 p2 p23)
        (abac p11 p23 p3 p34)
        (abac p11 p34 p4 p41)
        (command "_LAYER" "_S" l "")
      )
    )
    (r_fmi)
    (setq p1 (getpoint "\nABACO CENTRAL Primera esquina :")
          pp p1
    )
  )
  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


;*****************************************************************************
;* * *  ABACO DE BORDE
;*****************************************************************************

(defun c:ABACOB  (/ p1 p2 p3 p4 lab p12 p34 pp)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)

  (scpu)
  (r_fmi)

  (setq  p1 (getpoint "\nLADO del BORDE  Primera esquina :")
         pp p1
  )
  (while (/= pp nil)
```

```
    (if (/= pp nil)
      (progn
        (setq p2  (getpoint p1 "\nLADO del BORDE Segunda esquina :")
              p3  (getpoint p2 "\n               Tercera  esquina:")
              p4  (getpoint p3 "\n               Cuarta  esquina :")
              p12 (pmig p1 p2)
              p34 (pmig p3 p4)
        )
        (r_non)
        (setq l   (getvar "CLAYER")
              lrr (strcat l "-rec")
              lab (strcat l "-abac")
        )
        (command "_LAYER" "_N" lab "")
        (command "_LAYER" "_S" lab "")
        (abac p12 p2 p3 p34)
        (abac p12 p34 p4 p1)
        (command "_LAYER" "_S" l "")
      )
    )
    (r_fmi)
    (setq p1 (getpoint "\nLADO del BORDE Primera esquina :")
          pp p1
    )
  )
  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


;*****************************************************************************
;* * *  ABACO DE ESQUINA
;*****************************************************************************

(defun c:ABACOE  (/ p1 p2 p3 p4 lab pp)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)

  (scpu)
  (r_fmi)

  (setq  p1 (getpoint "\nindicar extremo de ESQUINA :")
         pp p1
  )
  (while (/= pp nil)
    (if (/= pp nil)
      (progn
        (setq p2 (getpoint p1 "\n          Segundo extremo :")
              p3 (getpoint p2 "\n          Tercer  extremo :")
              p4 (getpoint p3 "\n          Cuarto  extremo :")
        )
        (r_non)
        (setq l   (getvar "CLAYER")
              lrr (strcat l "-rec")
              lab (strcat l "-abac")
        )
        (command "_LAYER" "_N" lab "")
        (command "_LAYER" "_S" lab "")
        (abac p1 p2 p3 p4)
        (command "_LAYER" "_S" l "")
```

```
      )
    )
    (r_fmi)
    (setq p1 (getpoint "\nindicar extremo de ESQUINA :")
          pp p1
    )
  )
  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)


;*****************************************************************************
;* * *  GENERA LOS 3DACARA DE UN ABACO BASE EN FORJ RETICULAR
;*****************************************************************************

(defun abac (p1 p2 p3 p4 / x y z zy xy)

  (setq x  (puntint p1 p2 0.5)
        y  (puntint p2 p3 0.5)
        zy (puntint p3 p4 0.5)
        z  (puntint x zy 0.5)
  )
  (3_CARA x p2 y z)
  (3_CARA z y p3 zy)
  (setq y (puntint p1 p4 0.5))
  (3_CARA y z zy p4)
  (setq zy (puntint p1 z 0.5)
        xy (puntint p1 x 0.5)
  )
  (3_CARA xy x z zy)
  (setq x (puntint p1 y 0.5))
  (3_CARA x zy z y)
  (3_CARA p1 xy zy x)
)


;*****************************************************************************
;* * *   GENERA UN ZUNCHO EN BORDE DE RECUADRO RETICULAR CON ABACOS
;*****************************************************************************

(defun c:ZUNB (/ p1 p2 p11 p22 px py lrr fr f1r bb pp)

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)

  (scpu)
  (r_fmi)

  (setq fr 0.15
        p1 (getpoint "\nLADO RECUADRO    Primer extremo:")
        pp p1
  )
  (while (/= pp nil)
    (if (/= pp nil)
      (progn
        (setq p2 (getpoint p1 "\n                Segundo extremo:")
              bb fr
        )
        (princ "\n Abaco fraccion de la luz <")
        (princ bb)
```

```
        (princ "> ")
        (setq fr (getreal))
        (if (eq (eval fr) nil)
          (setq fr bb)
        )
        (setq f1r (- 1 fr))
        (r_non)
        (setq l   (getvar "CLAYER")
              lrr (strcat l "-zun")
        )
        (command "_LAYER" "_N" lrr "_S" lrr "")
        (setq p11 (puntint p1 p2 fr)
              p22 (puntint p1 p2 f1r)
              px  (puntint p1 p11 0.25)
              py  (puntint p1 p11 0.5)
        )
        (command "_LINE" p1 px "")
        (command "_LINE" px py "")
        (command "_LINE" py p11 "")
        (nnlin p11 p22 6)
        (setq px (puntint p22 p2 0.5)
              py (puntint p22 p2 0.75)
        )
        (command "_LINE" p22 px "")
        (command "_LINE" px py "")
        (command "_LINE" py p2 "")
        (command "_LAYER" "_S" l "")
      )
    )
    (r_fmi)
    (setq p1 (getpoint "\nLADO RECUADRO    Primer extremo:")
          pp p1
    )
  )
  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)
```

6.3 Módulo para describir Características Geométricas y Mecánicas de los elementos estructurales

```
; ************ FUNCIONES PARA GESTIONAR LA GEOMETRIA Y LOS MATERIALES EN EFCiD


; modest       ACTUALIZA EL TIPO DE ESTRUCTURA EN EL BLOQUE TIPEST
;
;
; insecrec     PIDE DATOS SECCION RECTANGULAR
; insechur     PIDE DATOS SECCION RECTANGULAR HUECA
; inseccir     PIDE DATOS SECCION CIRCULAR
; insechuc     PIDE DATOS SECCION CIRCULAR HUECA
; insecte      PIDE DATOS SECCION EN "T"
; insectef     PIDE DATOS SECCION EN "NERVIO EN T FORJADO UNIDIRECCIONAL
; insecter     PIDE DATOS SECCION EN "NERVIO EN T FORJADO RETICULAR
```

```
; insecphr     PIDE DATOS SECCION PHR
; insecphc     PIDE DATOS SECCION PHC
; insecph0     PIDE DATOS SECCION PH0
; insecipn     PIDE DATOS SECCION IPN
; insecipe     PIDE DATOS SECCION IPE
; insecheb     PIDE DATOS SECCION HEB
; insechea     PIDE DATOS SECCION HEA
; insechem     PIDE DATOS SECCION HEM
; insec2upn    PIDE DATOS SECCION 2UPN EN CAJON
; insecpergen  PIDE DATOS SECCION PERFIL GENERAL
; insecgen     PIDE DATOS SECCION GENERICA
; insecusr     FUNCION QUE PIDE DATOS SECCION DE USUARIO
;              Y PERFILES CONFORMADOS EN FRIO
; insecl
; inseclf
; inseczf
; insecld
; insecldf
; insecuf
; insecof
; inseccf
; insecupn
; secusr       SECCION cualquiera definida por el USUARIO
;
;
; inesp        PIDE ESPESOR DE PLACAS
;
;
; girocero     PONE A CERO EL ANGULO DE GIRO DE LA BARRA
; intgiro      INTRODUCE EL ANGULO DE GIRO DE LA BARRA
;
;
; asignar      ASIGNACION DEL PATRON DE ELEMENTO ACTIVO A UNO/VARIOS ELEMENTOS
;
;
; matgener     PROPIEDADES DE UN MATERIAL GENERICO
; insmater     INSERCION DEL BLOQUE TIPO DE MATERIAL
; testhormi    COMPRUEBA SI EL MATERIAL ACTUAL ES HORMIGON
; testacero    COMPRUEBA SI EL MATERIAL ACTUAL ES ACERO
; selmat       SELECCION DE UN MATERIAL
; acero        PROPIEDADES DEL ACERO
; hormigon     PROPIEDADES DEL HORMIGON
; madera       PROPIEDADES DE LA MADERA
; panal        PROPIEDADES DEL LADRILLO
; bloque       PROPIEDADES DE LA FABRICA DE BLOQUES DE HORMIGON
; termo        PROPIEDADES DE LA FABRICA DE BLOQUES DE TERMOARCILLA


(prompt "Cargando las utilidades de GEOMETRIA \n")


;*****************************************************************************
;* * * ACTUALIZA EL TIPO DE ESTRUCTURA EN EL BLOQUE TIPEST
;*****************************************************************************

(defun modest (tipest / conj esc n1 n2 n3 p tip)

  (->)
  (setq conj (ssget "X" (list (cons 2 "TIPEST"))))
  (if (/= nil conj)
    (if (> (sslength conj) 1)
      (prompt "Elimine los bloque TIPEST sobrantes ")
      (progn (setq n1  (ssname conj 0)
                   n2  (entnext n1)
                   n3  (entnext n2)
                   tip (entget n2)
```

```
                    esc (entget n3)
                    p   (cons 1 tipest)
                )
                (entmod (subst p (assoc 1 tip) tip))
                (setq p (cons 1 (rtos ef 2 2)))
                (entmod (subst p (assoc 1 esc) esc))
                (entupd n1)
        )
      )
    )
    (if (= tipest "CELOSIA PLANA")
      (progn (scpu)
             (EjeZ)
             (command "_PLAN" "")
      )
    )
    (if (= tipest "RIGIDA PLANA")
      (progn (scpu)
             (EjeZ)
             (command "_PLAN" "")
      )
    )
    (if (= tipest "EMPARRILLADO")
      (progn (scpu)
             (command "_VPOINT" "3,2,2")
      )
    )
    (if (= tipest "CELOSIA ESPACIAL")
      (progn (scpu)
             (command "_VPOINT" "3,2,2")
      )
    )
    (if (= tipest "RIGIDA ESPACIAL")
      (progn (scpu)
             (command "_VPOINT" "3,2,2")
      )
    )
  )


;*****************************************************************************
;* * * PIDE DATOS SECCION RECTANGULAR
;*****************************************************************************

(defun insecrec ( / b h)

   (->)
   (>?)

   (setq  b      (getreal "\nAncho seccion  cm:")
          h      (getreal "\nCanto seccion  cm:")
          nomsec "RECTANGULAR"
          dimsec (strcat "bxh "(rtos b 2 0) "x" (rtos h 2 0))
   )
   (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION RECTANGULAR HUECA
;*****************************************************************************

(defun insechur  (/ br hr er)

  (setq  br     (getreal "\nAncho seccion  cm:")
         hr     (getreal "\nCanto seccion  cm:")
```

```
        er     (getreal "\nEspesor seccion  cm:")
        nomsec "RECT-HUECA"
        dimsec (strcat "bxhxe "
                       (rtos br 2 0)
                       "x"
                       (rtos hr 2 0)
                       "x"
                       (rtos er 2 1)
               )
  )
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION CIRCULAR
;*****************************************************************************

(defun inseccir  (/ dc)

  (setq  dc     (getreal "\nDiametro seccion  cm:")
         nomsec "CIRCULAR"
         dimsec (strcat "D " (rtos dc 2 0))
  )
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION CIRCULAR HUECA
;*****************************************************************************

(defun insechuc  (/ d er)

  (setq  d      (getreal "\nDiametro seccion  cm:")
         er     (getreal "\nEspesor seccion  cm:")
         nomsec "CIRC-HUECA"
         dimsec (strcat "Dxe " (rtos d 2 0) "x" (rtos er 2 1))
  )
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION EN "T"
;*****************************************************************************

(defun insecte (/ br hr br1 hr1)

  (setq  br     (getreal "\nAncho secc.alas cm:")
         hr     (getreal "\nCanto total     cm:")
         br1    (getreal "\nAncho alma      cm:")
         hr1    (getreal "\nCanto alas      cm:")
         nomsec "TE"
         dimsec (strcat "BxHxalmaxala "
                        (rtos br 2 0)
                        "x"
                        (rtos hr 2 0)
                        "x"
                        (rtos br1 2 1)
                        "x"
                        (rtos hr1 2 1)
                )
  )
  (actualiza)
```

```
)

;*****************************************************************************
;* * * PIDE DATOS SECCION EN "NERVIO EN T FORJADO UNIDIRECCIONAL
;*****************************************************************************
(defun insectef  (/ br hr br1 hr1)

  (setq  br     (getreal "\nAncho secc.alas cm:")
         hr     (getreal "\nCanto total     cm:")
         br1    (getreal "\nAncho alma      cm:")
         hr1    (getreal "\nCanto alas      cm:")
         nomsec "NERVIO"
         dimsec (strcat "BxHxalmaxala "
                        (rtos br 2 0)
                        "x"
                        (rtos hr 2 0)
                        "x"
                        (rtos br1 2 1)
                        "x"
                        (rtos hr1 2 1)
                )
  )
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION EN "NERVIO EN T FORJADO RETICULAR
;*****************************************************************************
(defun insecter  (/ br hr br1 hr1)

  (setq  br     (getreal "\nAncho secc.alas cm:")
         hr     (getreal "\nCanto total     cm:")
         br1    (getreal "\nAncho alma      cm:")
         hr1    (getreal "\nCanto alas      cm:")
         nomsec "RETICULAR"
         dimsec (strcat "BxHxalmaxala "
                        (rtos br 2 0)
                        "x"
                        (rtos hr 2 0)
                        "x"
                        (rtos br1 2 1)
                        "x"
                        (rtos hr1 2 1)
                )
  )
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION PHR
;*****************************************************************************
(defun insecphr  (tip h e)

  (setq  nomsec "PHR"
         dimsec (strcat (rtos tip 2 0) "x" (rtos h 2 0) "x" (rtos e 2 0))
  )
  (testacero)
  (actualiza)
)
```

```
;*****************************************************************************
;* * * PIDE DATOS SECCION PHC
;*****************************************************************************

(defun insecphc  (tip e)

  (setq  nomsec "PHC"
         dimsec (strcat (rtos tip 2 0) "x" (rtos tip 2 0) "x" (rtos e 2 0))
  )
  (testacero)
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION PH0
;*****************************************************************************

(defun insecph0  (tip e)

  (setq  nomsec "PH0"
         dimsec (strcat (rtos tip 2 0) "x" (rtos e 2 0))
  )
  (testacero)
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION IPN
;*****************************************************************************

(defun insecipn  (tip)

  (setq  nomsec "IPN"
         dimsec (strcat (rtos tip 2 0))
  )
  (testacero)
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION IPE
;*****************************************************************************

(defun insecipe  (tip)

  (setq  nomsec "IPE"
         dimsec (strcat (rtos tip 2 0))
  )
  (testacero)
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION HEB
;*****************************************************************************

(defun insecheb  (tip)

  (setq  nomsec "HEB"
         dimsec (strcat (rtos tip 2 0))
  )
```

```
  (testacero)
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION HEA
;*****************************************************************************
'
(defun insechea  (tip)

  (setq  nomsec "HEA"
         dimsec (strcat (rtos tip 2 0))
  )
  (testacero)
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION HEM
;*****************************************************************************
'
(defun insechem  (tip)

  (setq  nomsec "HEM"
         dimsec (strcat (rtos tip 2 0))
  )
  (testacero)
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION 2UPN EN CAJON
;*****************************************************************************
'
(defun insec2upn (tip)

  (setq  nomsec "2UPN"
         dimsec (strcat (rtos tip 2 0))
  )
  (testacero)
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION PERFIL GENERAL
;*****************************************************************************
'
(defun insecpergen ()

  (setq  nomsec (getstring "\nNombre Perfil :")
         dimsec (getstring "\nNumero Perfil:")
  )
  (actualiza)
)


;*****************************************************************************
;* * * PIDE DATOS SECCION GENERICA
;*****************************************************************************
'
(defun insecgen  (/ ax ix iy iz)
```

```
  (setq  ax     (getreal "\nArea        cm2:")
         iz     (getreal "\nIz          cm4:")
         iy     (getreal "\nIy          cm4:")
         ix     (getreal "\nIx(tors)    cm4:")
         nomsec "GENERICA"
         dimsec (strcat "Ax "
                        (rtos ax 2 0)
                        " Ix "
                        (rtos ix 2 0)
                        " Iy "
                        (rtos iy 2 0)
                        " Iz "
                        (rtos iz 2 0)
                )
  )
  (actualiza)
)


;*****************************************************************************
;* * * FUNCION QUE PIDE DATOS SECCION DE USUARIO Y PERFILES CONFORMADOS EN FRIO
;*****************************************************************************

(defun insecusr  (tip n)

  (setq  nomsec "USUARIO"
         dimsec (strcat "_" tip "_" n)
  )
  (actualiza)
)


(defun insecl (a b)

  (testacero)
  (insecusr "L" (strcat (rtos a 2 0) "-" (rtos b 2 0)))
)


(defun inseclf (a b)

  (testacero)
  (insecusr "LF" (strcat (rtos a 2 0) "-" (rtos b 2 0)))
)


(defun inseczf (a b)

  (testacero)
  (insecusr "ZF"
            (strcat (rtos a 2 0) "-" (rtos (* 10 b) 2 0))
  )
)


(defun insecld (a b c)

  (testacero)
  (insecusr "LD"
            (strcat (rtos a 2 0) "-" (rtos b 2 0) "-" (rtos c 2 0))
  )
)


(defun insecldf  (a b c)
```

```
  (testacero)
  (insecusr "LDF"
            (strcat (rtos a 2 0) "-" (rtos b 2 0) "-" (rtos c 2 0))
  )
)


(defun insecuf (a b c)

  (testacero)
  (insecusr "UF"
            (strcat (rtos a 2 0) "-" (rtos b 2 0) "-" (rtos c 2 0))
  )
)


(defun insecof (a b c)

  (testacero)
  (insecusr "OF"
            (strcat (rtos a 2 0)
                    "-"
                    (rtos b 2 0)

                    "-"
                    (rtos (* 10 c) 2 0)
            )
  )
)


(defun inseccf (a b)

  (testacero)
  (insecusr "CF" (strcat (rtos a 2 0) "-" (rtos b 2 0)))
)


(defun insecupn  (a)

  (testacero)
  (insecusr "UPN" (rtos a 2 0))
)



;*****************************************************************************
;* * * SECCION cualquiera definida por el USUARIO
;*****************************************************************************

(defun secusr (/ fich secc)

  (setq  fich (getstring "\nDenominacion del fichero   :")
         secc (getstring "\nDescripción de la sección  :")
  )
  (insecusr fich secc)
)


;*****************************************************************************
;* * * PIDE ESPESOR DE PLACAS
;*****************************************************************************

(defun inesp ()
```

```
   (setq nomsec "Placa"
         esplac (getreal "\nEspesor de placa cm:")
          dimsec (strcat "Espesor " (rtos (* 10 esplac) 2 0))
   )
   (actualiza)
)


;*****************************************************************************
;* * * PONE A CERO EL ANGULO DE GIRO DE LA BARRA
;*****************************************************************************

(defun girocero (  / )

   (setq anggiro 0)
   (actualiza)
)


;*****************************************************************************
;* * * INTRODUCE EL ANGULO DE GIRO DE LA BARRA
;*****************************************************************************

(defun intgiro (  / )

   (setq anggiro (getreal "\nAngulo giro(º) :"))
   (actualiza)
)


;*****************************************************************************
;* * * ASIGNACION DEL PATRON DE ELEMENTO ACTIVO A UNO O VARIOS ELEMENTOS
;*****************************************************************************

(defun asignar (tip / ent entac cap conj v n fich tl tll)

  (noecho)

  (diano)

  (cond
    ((= 1 tip)
     (setq tl  (props->tl)
           tll (cons 6 tl)
     )
    )
    ((and (= 2 tip) (= nomsec "Placa"))
     (setq tl  (strcat "M"
                       (chr (+ 64 (atoi nummat)))
                       "0I"
                       (rtos (* esplac 10) 2 0)
                 )
           tll (cons 6 tl)
     )
    )
    ((and (= 3 tip) (= nomsec "Solido"))
     (setq tl  (strcat "N" (chr (+ 64 (atoi nummat))) "0I")
           tll (cons 6 tl)
     )
    )
    (T (setq tl ""))
  )
  (if (= tl "")
    (progn (prompt "\nNo es posible asignar las propiedades activas.")
           (prompt "\nSe trata de elementos de distinto tipo.")
    )
```

```
    (progn
      (if (= nil (tblsearch "LTYPE" tl))
        (progn (CreaTl tl)
               (CargaTl tl)
        )
      )
      (setq fich (open "c:/cid/cad/st.lin" "w"))
      (close fich)
      (while (not (setq conj (ssget))))
      (setq v 0)
      (repeat (sslength conj)
        (setq ent (ssname conj v)
              n   (entget ent)
        )
        (cond ((and (= tip 1) (= "LINE" (cdr (assoc 0 n))))
               (setq cap (assoc 6 n))
               (if (= nil cap)
                  (setq entac (cons tll n))
                  (setq entac (subst tll cap n))
               )
               (entmod entac)
              )
              ((and (= tip 2) (= "3DFACE" (cdr (assoc 0 n))))
               (setq cap (assoc 6 n))
               (if (= nil cap)
                  (setq entac (cons tll n))
                  (setq entac (subst tll cap n))
               )
               (entmod entac)
              )
              ((and (= tip 3) (= "POLYLINE" (cdr (assoc 0 n))))
               (setq cap (assoc 6 n))
               (if (= nil cap)
                  (setq entac (cons tll n))
                  (setq entac (subst tll cap n))
               )
               (entmod entac)
              )

        )
        (setq v (+ v 1))
      )
    )
  )

)


;*****************************************************************************
;
;* * * PROPIEDADES DE UN MATERIAL GENERICO
;*****************************************************************************
;

(defun matgener ( / n)

       (setq  nommat (getstring "\nNombre del Material                  :"))
       (setq  module (getreal   "\nModulo de Young E              (Kp/cm2):"))
       (setq  poiss  (getreal   "\nCoeficiente de Poisson                :"))
       (setq  dens   (getreal   "\nPeso especifico                (Kp/m3) :"))
       (setq  cterm  (getreal   "\nCoeficiente Dilataci¢n Termica         :"))
       (setq n (+ (getvar "useri2") 1))
       (setvar "useri2" n)
       (setq nummater (strcat (rtos n 2 0) " " nommat))
       (setq nummat n)
       (insmater "GENERICO")
)
```

```
;*****************************************************************************
;* * * INSERCION DEL BLOQUE TIPO DE MATERIAL
;*****************************************************************************

(defun insmater ( m / ent l mat n p p1 p2 p3 p4 p5 p6 n1 n2 n3 n4 n5 n6)

   (cpscp)
   (EjeZ)
   (cpcap)
   (setq l (getvar "CLAYER"))
   (if (not (wcmatch l "TIPOMATS"))
       (command "_LAYER" "_T" "TIPOMATS" "_ON" "TIPOMATS" "_S" "TIPOMATS" "")
   )
   (setq p (getpoint (strcat "\n Punto de inserción del MATERIAL " m)))
   (command "_INSERT" "MATERIAL" "_SC" 3 p "0.0")
   (pgcap)

   (setq mat (entlast)
         n1  (entnext mat)
         n2  (entnext n1)
         n3  (entnext n2)
         n4  (entnext n3)
         n5  (entnext n4)
         n6  (entnext n5)
         p1  (entget n1)                ; MATER  nom material   :: nommat
         p2  (entget n2)                ; NUMERO nº de material :: nummat
         p3  (entget n3)                ; E modulo elastic      :: module
         p4  (entget n4)                ; POISS  coef.Poisson   :: poiss
         p5  (entget n5)                ; PESP  peso especifico :: dens
         p6  (entget n6))               ; CTERM coef.dilatacion :: cterm

   (setq p (cons 1 nommat))               (entmod (subst p (assoc 1 p1) p1))
   (setq v (rtos nummat 2 0) p (cons 1 v)) (entmod (subst p (assoc 1 p2) p2))
   (setq v (rtos module 2 0) p (cons 1 v)) (entmod (subst p (assoc 1 p3) p3))
   (setq v (rtos poiss  2 2) p (cons 1 v)) (entmod (subst p (assoc 1 p4) p4))
   (setq v (rtos dens   2 0) p (cons 1 v)) (entmod (subst p (assoc 1 p5) p5))
   (setq v (rtos cterm  2 6) p (cons 1 v)) (entmod (subst p (assoc 1 p6) p6))

   (entupd mat)

   (pgscp)
)


;*****************************************************************************
;* * * COMPRUEBA SI EL MATERIAL ACTUAL ES HORMIGON
;*****************************************************************************

(defun testhormi (/ b m mat)

  (setq  b   (ssname (ssget "X" (list (cons 2 "PATACT"))) 0)
         m   (entnext b)
         mat (cdr (assoc 1 (entget m)))
  )
  (if (wcmatch mat "~*HORMIGON*")
    (progn (prompt "El material seleccionado no es HORMIGON. ")
           (prompt "\nSeleccionelo ahora")
           (C:SS)
    )
  )
)


;*****************************************************************************
```

```
;* * * COMPRUEBA SI EL MATERIAL ACTUAL ES ACERO
;*****************************************************************************
(defun testacero (/ b m mat ok)

  (setq b   (ssname (ssget "X" (list (cons 2 "PATACT"))) 0)
        m   (entnext b)
        mat (cdr (assoc 1 (entget m)))
  )
  (if (wcmatch mat "~*ACERO*")
    (progn (prompt "El material seleccionado no es ACERO. ")
           (setq ok
                  (getstring
                    "\n¿Desea seleccionar el material  2 ACERO ?  <S> "
                  )
           )
           (if (= ok "")
             (progn (setq nummater "2 ACERO")
                    (actualiza)
             )
             (C:SS)
           )
    )
  )
)


;*****************************************************************************
;* * * SELECCION DE UN MATERIAL
;*****************************************************************************
(defun selmat (mat / p1 p2 p3 p4 p5 p6 n1 n2 n3 n4 n5 n6)

  (->)
  (setq n1       (entnext mat)
        n2       (entnext n1)
        p1       (entget n1)
        p2       (entget n2)
        nommat   (cdr (assoc 1 p1))
        nummat   (cdr (assoc 1 p2))
        nummater (strcat nummat " " nommat)
  )
  (actualiza)
)


;*****************************************************************************
;* * * PROPIEDADES DEL ACERO
;*****************************************************************************
(defun acero (/ n)

  (setq nommat "ACERO"
        module 2100000
        poiss  0.3
        dens   7850
        cterm  0.000012
        n      (+ (getvar "useri2") 1)
  )
  (setvar "useri2" n)
  (setq nummater (strcat (rtos n 2 0) " " nommat)
        nummat   n
  )

  (insmater "ACERO")
)
```

```
;*****************************************************************************
;* * * PROPIEDADES DEL HORMIGON
;*****************************************************************************

(defun hormigon  (/ n)

  (setq  nommat "HORMIGON"
         module 250000
         poiss  0.2
         dens   2500
         cterm  0.00001
         n      (+ (getvar "useri2") 1)
  )
  (setvar "useri2" n)
  (setq  nummater (strcat (rtos n 2 0) " " nommat)
         nummat   n
  )

  (insmater "HORMIGON")
)


;*****************************************************************************
;* * * PROPIEDADES DE LA MADERA
;*****************************************************************************

(defun madera (/ n)

  (setq  nommat "MADERA"
         module 120000
         poiss  0.15
         dens   500
         cterm  0.00001
         n      (+ (getvar "useri2") 1)
  )
  (setvar "useri2" n)
  (setq  nummater (strcat (rtos n 2 0) " " nommat)
         nummat   n
  )
  (insmater "MADERA")
)


;*****************************************************************************
;* * * PROPIEDADES DEL LADRILLO
;*****************************************************************************

(defun panal (/ n)

  (setq  nommat "LADRILLO"
         module 70000
         poiss  0.2
         dens   1600
         cterm  0.00001
         n      (+ (getvar "useri2") 1)
  )
  (setvar "useri2" n)
  (setq  nummater (strcat (rtos n 2 0) " " nommat)
         nummat   n
  )
  (insmater "LADRILLO")
)
```

```
;*****************************************************************************
;* * * PROPIEDADES DE LA FABRICA DE BLOQUES DE HORMIGON
;*****************************************************************************

(defun bloque (/ n)

  (setq  nommat "BLOQUE"
         module 80000
         poiss  0.2
         dens   1500
         cterm  0.00001
         n      (+ (getvar "useri2") 1)
  )
  (setvar "useri2" n)
  (setq  nummater (strcat (rtos n 2 0) " " nommat)
         nummat   n
  )
  (insmater "BLOQUE")
)


;*****************************************************************************
;* * * PROPIEDADES DE LA FABRICA DE BLOQUES DE TERMOARCILLA
;*****************************************************************************

(defun termo (/ n)

  (setq  nommat "TERMOARCILLA"
         module 50000
         poiss  0.2
         dens   1500
         cterm  0.00001
         n      (+ (getvar "useri2") 1)
  )
  (setvar "useri2" n)
  (setq  nummater (strcat (rtos n 2 0) " " nommat)
         nummat   n
  )
  (insmater "TERMOARCILLA")
)
```

6.4 Módulo para obtener las propiedades Mecánicas de una sección y la distribución de tensiones normales

```
;*****************************************************************************

; Obtenci¢n de las propiedades geométricas y tensiones en una sección arbitraria
;*****************************************************************************


; ?cero       FUNCIONES BASICAS
; siH
; noH
; noecho
; diasi
; diano
; r_non
; r_varios
; getconj
```

```
; creabi      Crea una REGION ABIERTA a partir de una polilínea
; crealv      Crea una REGION ALVEOLADA a partir de un conjunto de polilineas
; prop_reg    Propiedades mecánicas de una REGION respecto al Sist. Referencia Actual
; cuadro      Inserta un bloque que muestra las propiedades mecánicas de la REGION
; C:PROP
; C:PROPCDG
; scp_cdg     Cambia el origen del SCP al CDG de la REGION y obtiene Prop. Mecánicas
; C:CDG
; ejes_ppales Cambia al SCP de direcciones principales e inserta bloque ejes
orientados
; C:PPAL
; erg         Dibuja la elipse de los radios de giro centrada
; nmm         Obtención de las solicitaciones sobre la sección
; C:NMM
; nee
; C:NEE
; solicit     Inserci¢n del cuadro de solicitaciones
; coef        Coeficientes de la ecuación de tensiones
; zona        Zona dentro de la que se dibuja el bloque de PROPIEDADES
; C:ZON
; zonaEN      Zona dentro de la que se dibujan el EJE NEUTRO o el NUCLEO CENTRAL
; C:ZONEN
; ejen        Trazado del EJE NEUTRO
; C:EN
; nucleo      Trazado del NUCLEO CENTRAL
; C:NC
; tension     Calcula la tensi¢n en un punto e inserta un bloque
; C:TEN
; mov_en      Desplaza el eje neutro según sea el punto de aplicacion del axil
; C:MEN



;*****************************************************************************
;* * * INICIALIZACIONES
;*****************************************************************************


(setq +cero    0.000001
      -cero   -0.000001
      escblk   1.0
)
(setvar "MIRRTEXT" 1)
(setvar "UCSICON" 0)

(command "_LAYER" "_N"          "REGION,SOMBREADO"
                  "_T"          "0,REGION,SOMBREADO"
                  "_ON"         "0,REGION"
                  "_S"          "0"
                  "_COLOR" "30" "REGION"
                  "_COLOR" "31" "SOMBREADO" ""
)

(prompt "\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n ")



;*****************************************************************************
;* * * FUNCIONES BASICAS
;*****************************************************************************

(defun ?cero (n)

  (and (> n -cero) (< n +cero))
)
```

```
(defun siH ()

  (setvar "HIGHLIGHT" 1)
)

(defun noH ()

  (setvar "HIGHLIGHT" 0)
)

(defun noecho ()

  (setvar "CMDECHO" 0)
)

(defun diasi ()

  (setvar "ATTDIA" 1)
)


(defun diano ()

  (setvar "ATTDIA" 0)
)

(defun r_non ()

  (setvar "OSMODE" 0)
)

(defun r_varios  ()

  (setvar "OSMODE" 1195)
)

(defun getconj (msg)

  (prompt msg)
  (while (not (setq conj (ssget))))
)


;*****************************************************************************
;* * * Crea una REGION ABIERTA a partir de una polilínea
;*****************************************************************************

(defun creabi (/ cl)

  (noecho)
  (r_non)
  (siH)

  (setq cl (getvar "CLAYER"))
  (command "_LAYER" "_ON" "REGION" "_OFF" "SOMBREADO" "")
  (setvar "CLAYER" "REGION")

  (getconj "\nDesigne el contorno de la seccion ")

  (if (= (sslength conj) 1)
    (progn (command "_REGION" conj "")
           (setq secc (entlast))
    )
    (progn (command "_REGION" (ssname conj 0) "")
           (setq secc (entlast)
                 t0   1
```

```
          )
          (repeat (- (sslength conj) 1)
            (command "_REGION" (ssname conj t0) "")
            (command "_UNION" secc (entlast) "")
            (setq secc  (entlast)
                  t0   (+ t0 1)
            )
          )
    )
  )
  (setvar "CLAYER" "SOMBREADO")
  (command "_HATCH" "_S" secc "")
  (command "_LAYER" "_ON" "SOMBREADO" "")
  (noH)
  (setvar "CLAYER" cl)
)


;*****************************************************************************
;* * * Crea una REGION ALVEOLADA a partir de un conjunto de polilineas
;*****************************************************************************

(defun crealv
              (/ cl                     ; Nombre de la capa actual
               cex                      ; Región con el contorno exterior
               alv                      ; Región con los alveolos
               t0                       ; Contador auxiliar
)


  (noecho)
  (r_non)
  (siH)

  (setq cl (getvar "CLAYER"))
  (command "_LAYER" "_ON" "REGION" "_OFF" "SOMBREADO" "")
  (setvar "CLAYER" "REGION")

  (getconj
    "\n\nSeleccione el contorno exterior de la seccion "
  )

  (if (= (sslength conj) 1)
    (progn (command "_REGION" conj "")
          (setq cex (entlast))
    )
    (progn (command "_REGION" (ssname conj 0) "")
          (setq cex (entlast)
                t0  1
          )
          (repeat (- (sslength conj) 1)
            (command "_REGION" (ssname conj t0) "")
            (command "_UNION" cex (entlast) "")
            (setq cex (entlast)
                  t0  (+ t0 1)
            )
          )
    )
  )

  (getconj
    "\n\nSeleccione el contorno del hueco de la seccion "
  )

  (if (= (sslength conj) 1)
    (progn (command "_REGION" conj "")
```

```
            (setq alv (entlast))
    )
    (progn (command "_REGION" (ssname conj 0) "")
            (setq alv (entlast)
                  t0  1
            )
            (repeat (- (sslength conj) 1)
              (command "_REGION" (ssname conj t0) "")
              (command "_UNION" alv (entlast) "")
              (setq alv (entlast)
                    t0  (+ t0 1)
              )
            )
    )
  )

  (command "_SUBTRACT" cex "" alv "")

  (setq secc (entlast))
  (setvar "CLAYER" "SOMBREADO")
  (command "_HATCH" "_S" secc "")
  (command "_LAYER" "_ON" "SOMBREADO" "")

  (noH)
  (setvar "CLAYER" cl)
)


;*****************************************************************************
;* * * Propiedades mecánicas de una REGION respecto al Sist. Referencia Actual
;*****************************************************************************

(defun prop_reg  ()

  (noecho)
  (command "_MASSPROP" secc "" "_Y" "c:/props")
  (prompt
    "\n\n\n\nPropiedades fisicas de la region respecto del SCP actual"
  )
  (if (/= fil nil)
    (close fil)
  )
  (setq preg "c:/props.mpr")
  (if (/= preg nil)
    (progn (setq fil (open preg "r"))
            (read-line fil)
            (read-line fil)
            (read-line fil)
            (setq A (atof (substr (read-line fil) 26)))
            (read-line fil)
            (read-line fil)
            (read-line fil)
            (setq zcdg (atof (substr (read-line fil) 36))
                  ycdg (atof (substr (read-line fil) 26))
                  I_z  (atof (substr (read-line fil) 27))
                  I_y  (atof (substr (read-line fil) 26))
                  Izy  (atof (substr (read-line fil) 27))
                  Rz   (atof (substr (read-line fil) 23))
                  Ry   (atof (substr (read-line fil) 26))
            )
            (read-line fil)
            (setq I1V1 (substr (read-line fil) 26)
                  I2V2 (substr (read-line fil) 26)
                  v    1
                  ca   "W"
            )
```

```
            (while (/= ca " ")
              (setq ca (substr I1V1 v 1)
                    v  (1+ v)
              )
            )
            (setq I1 (atof (substr I1V1 1 v))
                  v  1
                  ca "W"
            )
            (while (/= ca "[")
              (setq ca (substr I1V1 v 1)
                    v  (1+ v)
              )
            )
            (setq V1 (substr I1V1 v)
                  V1 (substr V1 1 (1- (strlen V1)))
            )
            (setq v  1
                  ca "W"
            )
            (while (/= ca " ")
              (setq ca (substr I2V2 v 1)
                    v  (1+ v)
              )
            )
            (setq I2 (atof (substr I2V2 1 v))
                  v  1
                  ca "W"
            )
            (while (/= ca "[")
              (setq ca (substr I2V2 v 1)
                    v  (1+ v)
              )
            )
            (setq V2 (substr I2V2 v)
                  V2 (substr V2 1 (1- (strlen V2)))
            )
            (close fil)
    )
  )
)


;*****************************************************************************
;* * * Inserta un bloque que muestra las propiedades mecánicas de la REGION
;*****************************************************************************

(defun cuadro
               (/ p                       ; Punto de inserci¢n del cuadro resumen
                tA                        ; Texto del area de la REGION
                tI_z                      ;   "    "  momento de inercia   Iz
                tI_y                      ;   "    "  momento de inercia   Iy
                tIzy                      ;   "    "  producto de inercia  Izy
                tRz                       ;   "    "  radio de giro        Rz
                tRy                       ;   "    "  radio de giro        Ry
                tV1                       ;   "    "  vector eje principal V1
                tV2                       ;   "    "  vector eje principal V2
                tI1                       ;   "    "  momento de inercia   I1
                tI2                       ;   "    "  momento de inercia   I2
                c1 c2 nt                  ; Tipo de notación a utilizar
                d                 ; Precisión de la notación
)
  (noecho)
  (noH)
  (diano)
  (r_non)
```

```
  (command "_UCS" "")
  (zona)
  (setq escblk (abs (/ (distance p1 p2) 22)))
  (cond ((< escblk 0.2)
         (setq nt 2
               d  5
         )
        )
        ((< escblk 2)
         (setq nt 2
               d  2
         )
        )
        (T
         (setq nt 1
               d  3
         )
        )
  )
  (setq  tA   (rtos A nt d)
         tI_z (rtos I_z nt d)
         tI_y (rtos I_y nt d)
         tIzy (rtos Izy nt d)
         tRz  (rtos Rz 2 2)
         tRy  (rtos Ry 2 2)
         tV1  V1
         tV2  V2
         tI1  (rtos I1 nt d)
         tI2  (rtos I2 nt d)
  )

  (command "_INSERT" "COPYRIGHT" p1 escblk escblk "0.0")
  (command "_INSERT"         "PROPS" p1       escblk  escblk  "0.0"
           tA       tI_z     tI_y     tIzy     tRz      tRy     tV1
           tV2      tI1      tI2
          )
  (command "_UCS" "_P")
  (diasi)
  (siH)
)


(defun C:PROP
               (
               )
  (siH)
  (command "_LAYER" "_ON" "REGION" "_OFF" "SOMBREADO" "")
  (command "_UCS" "_V")
  (setvar "UCSICON" 1)
  (setq  ori  (getpoint
                "\n\n\n\nDesigne el origen del nuevo Sistema de Referencia "
              )
         ejex (/  (* 180
                     (getorient ori "\nIndique la orientación del eje OX ")
                  )
                  pi
              )
  )
  (command "_UCS" "_M" ori)
  (command "_UCS" "_Z" ejex)
  (setq secc (car (entsel "\nSeleccione una región. ")))
  (prop_reg)
  (cuadro)
  (setq sistref "OTRO")
  (command "_LAYER" "_ON" "REGION,SOMBREADO" "")
```

```
  (prompt
    "\n\n\n\nPropiedades fisicas de la region calculadas con respecto al SCP actual"
  )
  (command "_UCS" "_V")
)


(defun C:PROPCDG
                  (
                  )
  (siH)
  (command "_LAYER" "_ON" "REGION" "_OFF" "SOMBREADO" "")
  (scp_cdg)
  (prop_reg)
  (cuadro)
  (command "_LAYER" "_ON" "REGION,SOMBREADO" "")
  (prompt
    "\n\n\n\nPropiedades fisicas de la region calculadas con respecto al SCP actual"
  )

)


;*****************************************************************************
;* * * Cambia el origen del SCP al CDG de la REGION y obtiene Prop. Mecánicas
;*****************************************************************************

(defun scp_cdg (/ orig om o c)

  (noecho)
  (setq om (getvar "OSMODE"))
  (r_non)
  (noH)
  (command "_LAYER" "_ON" "REGION" "_OFF" "SOMBREADO" "")
  (setq secc (car (entsel "\nSeleccione una región. ")))
  (setvar "UCSICON" 0)
  (command "_UCS" "")
  (prop_reg)
  (setq  orig (list zcdg ycdg)
         o    (list 0 0)
         c    (ssget "X" (list (cons 2 "SISTREF")))
  )
  (command "_UCS" "_M" orig)
  (command "_UCS" "_X" "180.0")
  (prop_reg)
  (r_non)
  (if (/= c nil)
    (command "_ERASE" c "")
  )
  (command "_INSERT" "SISTREF" o escblk escblk "0.0")
  (setq sistref "YZCDG")
  (setvar "OSMODE" om)
  (siH)
)


(defun C:CDG ()

  (scp_cdg)
)


;*****************************************************************************
;* * * Cambia al SCP de direcciones principales e inserta bloque ejes orientados
;*****************************************************************************
```

```
(defun ejes_ppales
                    (/ c  cl              ; Capa actual al iniciar la rutina
                     o                    ; Origen del Sist. Referencia Principal
                     om                   ; Valor de OSMODE
                     e1                   ; Direcci¢n Principal 1
                     e2                   ; Direcci¢n Principal 2
                     v ca)

  (noecho)
  (setq om (getvar "OSMODE"))
  (r_non)
  (noH)
  (command "_LAYER" "_ON" "REGION" "_OFF" "SOMBREADO" "")
  (setq secc (car (entsel "\nSeleccione una región. ")))
  (setvar "UCSICON" 0)
  (command "_UCS" "")
  (prop_reg)
  (setq orig (list zcdg ycdg))
  (command "_UCS" "_M" orig)
  (command "_UCS" "_X" "180.0")
  (prop_reg)
  (setq v  1
        ca "W"
        e1 V1
  )
  (while (/= ca " ")
    (setq ca (substr e1 v 1)
          v  (1+ v)
    )
  )
  (setq e11 (atof (substr e1 1 v))
        e12 (atof (substr e1 v (strlen e1)))
  )
  (setq v  1
        ca "W"
        e2 V2
  )
  (while (/= ca " ")
    (setq ca (substr e2 v 1)
          v  (1+ v)
    )
  )
  (setq e21 (atof (substr e2 1 v))
        e22 (atof (substr e2 v (strlen e2)))
  )
  (setq c  (ssget "X" (list (cons 2 "PPALES")))
        cl (getvar "CLAYER")
        om (getvar "OSMODE")
        o  (list 0 0)
        e1 (list e11 e12)
        e2 (list e21 e22)
  )
  (r_non)
  (command "_UCS" "_3P" o e2 e1)
  (command "_LAYER" "_ON" "PPALES" "")
  (setvar "CLAYER" "PPALES")
  (if (/= c nil)
    (command "_ERASE" c "")
  )
  (command "_INSERT"
           "PPALES"
           o
           (* 3 escblk)
           (* 3 escblk)
           "0.0"
```

```
  )
  (setq sistref "PPALESCDG")
  (setvar "OSMODE" om)
  (setvar "CLAYER" cl)
  (siH)
)


(defun C:PPAL ()

  (ejes_ppales)
)


;*****************************************************************************
;* * * Dibuja la elipse de los radios de giro centrada
;*****************************************************************************

(defun erg (/ c cl om p1 p2)

  (noecho)
  (noH)
  (ejes_ppales)
  (prop_reg)
  (setq  p1 (list Ry 0)
         p2 (list (- Ry) 0)
         om (getvar "OSMODE")
         cl (getvar "CLAYER")
  )
  (r_non)
  (command "_LAYER" "_ON" "PPALES" "")
  (setvar "CLAYER" "PPALES")
  (setq c (ssget "X" (list (cons 8 "PPALES") (cons 0 "POLYLINE"))))
  (if (/= c nil)
    (command "_ERASE" c "")
  )
  (command "_ELLIPSE" p1 p2 Rz)
  (setvar "OSMODE" om)
  (setvar "CLAYER" cl)
  (siH)
)


;*****************************************************************************
;* * * Obtención de las solicitaciones sobre la sección
;*****************************************************************************

(defun nmm ()

  (noecho)
  (scp_cdg)
  (setq  N  (getreal "\nAxil   (en  kg) ")
         Mz (getreal "\nMomento según el eje OZ   (en  kg.cm)")
         My (getreal "\nMomento según el eje OY   (en  kg.cm)")
         ez 0
         ey 0
  )
  (solicit (rtos N 2 0)
           (rtos Mz 2 0)
           (rtos My 2 0)
           (rtos ez 2 2)
           (rtos ey 2 2)
  )
)
```

```
(defun C:NMM ()

  (nmm)
)


(defun nee (/ p)

  (noecho)
  (r_non)
  (scp_cdg)
  (setq  N (getreal "\nAxil   (en  kg) ")
         p (getpoint "\nSeleccione el punto de aplicación del axil ")
  )
  (if (= p nil)
    (setq ez (getreal "\nExcentricidad en la dirección Z   (en  cm)")
          ey (getreal "\nExcentricidad en la dirección Y   (en  cm)")
    )
    (setq ez (car p)
          ey (cadr p)
    )
  )
  (setq  Mz (* N ey)
         My (* N ez)
  )
  (solicit (rtos N 2 0)
           (rtos Mz 2 0)
           (rtos My 2 0)
           (rtos ez 2 2)
           (rtos ey 2 2)
  )
)


(defun C:NEE ()

  (nee)
)


;*****************************************************************************
;* * * Inserci¢n del cuadro de solicitaciones
;*****************************************************************************

(defun solicit (tN tMz tMy tez tey / p)   ; Punto inserci¢n cuadro solicitaciones

  (noecho)
  (noH)
  (r_non)
  (command "_UCS" "")
  (setq c (ssget "X" (list (cons 2 "SOLICIT"))))
  (if (/= c nil)
    (command "_ERASE" c "")
  )
  (command "_REDRAW")
  (setq p (getpoint "\nPosicion del cuadro de solicitaciones "))
  (diano)
  (command "_INSERT" "SOLICIT" p escblk   escblk "0.0" tN  tMz tMy  tez tey)
  (command "_UCS" "_P")
  (diasi)
  (siH)
)


;*****************************************************************************
```

```
;* * * Coeficientes de la ecuación de tensiones
;*****************************************************************************

(defun coef ()

  (noecho)
  (if (OR (= N nil) (= Mz nil) (= My nil))
    (progn (prompt
             "\nNo se han especificado por completo las solicitaciones "
           )
           (nmm)
           (coef)
    )
    (progn (setq det (- (* I_z I_y) (* Izy Izy))
                   f1  (/ (- (* I_z My) (* Izy Mz)) det)
                   f2  (/ (- (* I_y Mz) (* Izy My)) det)
                   f3  (/ N A)
           )
           (if (?cero f1)
             (setq f1 0)
           )
           (if (?cero f2)
             (setq f2 0)
           )
           (if (?cero f3)
             (setq f3 0)
           )
    )
  )
)


;*****************************************************************************
;* * * Zona dentro de la que se dibuja el bloque de PROPIEDADES
;*****************************************************************************

(defun zona (/)

  (prompt (strcat "\nZona en la que se dibuja el bloque"))
  (setq  p1 (getpoint "\nPrimera esquina ")
         p2 (getpoint "\nSegunda esquina ")
  )
  (prompt "\n\n")
)



(defun C:ZON ()

  (zona)
)


;*****************************************************************************
;* * * Zona dentro de la que se dibujan el EJE NEUTRO o el NUCLEO CENTRAL
;*****************************************************************************

(defun zonaEN (/)

  (prompt (strcat "\nZona en la que se dibuja el Eje Neutro"))
  (setq  p3 (getpoint "\nPrimera esquina ")
         p4 (getpoint "\nSegunda esquina ")
  )
  (prompt "\n\n")
)
```

```
(defun C:ZONEN ()

  (zonaEN)
)


;*****************************************************************************
;* * * Trazado del EJE NEUTRO
;*****************************************************************************
;

(defun ejen
            (/ z1 z2 y1  y2 ei ef eje)

  (noecho)
  (if (/= sistref "YZCDG")
    (scp_cdg)
  )
  (coef)

  (if (= f1 0)
    (if (= f2 0)
      (if (>= f3 0)
        (progn (prompt "\nEl estado es de tracción simple          ")
               (setq eje 0)
        )
        (progn (prompt "\nEl estado es de compresión simple         ")
               (setq eje 0)
        )
      )
      (setq z1 (car p3)
            y1 (- (/ f3 f2))              ; Recta horizontal
            z2 (car p4)
            y2 y1
            ei (list z1 y1)
            ef (list z2 y2)
      )
    )
    (if (= f2 0)
      (setq z1 (- (/ f3 f1))
            y1 (cadr p1)          ; Recta vertical
            z2 z1
            y2 (cadr p2)
            ei (list z1 y1)
            ef (list z2 y2)
      )
      (setq z1 (car p3)
            y1 (- (/ (+ f3 (* z1 f1)) f2))
            z2 (car p4)
            y2 (- (/ (+ f3 (* z2 f1)) f2))
            ei (list z1 y1)
            ef (list z2 y2)
      )
    )
  )
  (if (/= eje 0)
    (command "_LINE" ei ef "")
  )
)


(defun C:EN ()

  (zonaEN)
  (ejen)
)
```

```
;*****************************************************************************
;* * * Trazado del NUCLEO CENTRAL
;*****************************************************************************

(defun nucleo (/ p)

  (setq  p (getpoint
             "\nSeleccione un punto de la envolvente de la sección"
           )
  )
  (if (/= p nil)
    (progn (setq ez (car p)
                 ey (cadr p)
                 N  10000000.0
                 Mz (* N ey)
                 My (* N ez)
           )
           (ejen)
           (nucleo)
    )
  )
)


(defun C:NC (/ c cl)

  (setq  c  (ssget "X" (list (cons 2 "SOLICIT")))
         cl (getvar "CLAYER")
  )
  (if (/= c nil)
    (command "_ERASE" c "")
  )
  (command "_REDRAW")
  (command "_LAYER" "_OFF" "REGION" "_ON" "NC" "")
  (setvar "CLAYER" "NC")

  (scp_cdg)
  (zonaEN)
  (r_varios)
  (nucleo)

  (setvar "CLAYER" cl)
  (command "_LAYER" "_ON" "REGION" "")
  (setq  N  nil
         Mz nil
         My nil
  )
)


;*****************************************************************************
;* * * Calcula la tensi¢n en un punto e inserta un bloque
;*****************************************************************************

(defun tension
                (/ p                        ; Punto donde se calcularа la tension
                  pos                       ; Posición punto respecto de la REGION
                  z y sigma s)

  (noecho)
  (if (/= sistref "YZCDG")
    (scp_cdg)
  )
```

```
  (coef)

  (setq p (getpoint "Seleccione un punto "))
  (setq  z     (car p)
         y     (cadr p)
         sigma (+ f3 (* z f1) (* y f2))
         s     (rtos sigma 2 0)
  )
  (diano)

  (if (< sigma 0)
    (if (>= z 0)
      (if (>= y 0)
        (command "_INSERT" "COMPR1" p escblk escblk "0.0" s)
        (command "_INSERT" "COMPR2" p escblk escblk "0.0" s)
      )
      (if (>= y 0)
        (command "_INSERT" "COMPR4" p escblk escblk "0.0" s)
        (command "_INSERT" "COMPR3" p escblk escblk "0.0" s)
      )
    )
    (if (>= z 0)
      (if (>= y 0)
        (command "_INSERT" "TRACC1" p escblk escblk "0.0" s)
        (command "_INSERT" "TRACC2" p escblk escblk "0.0" s)
      )
      (if (>= y 0)
        (command "_INSERT" "TRACC4" p escblk escblk "0.0" s)
        (command "_INSERT" "TRACC3" p escblk escblk "0.0" s)
      )
    )
  )

  (diasi)
)


(defun C:TEN ()

  (tension)
)


;*****************************************************************************
;* * * Desplaza el eje neutro según sea el punto de aplicacion del axil
;*****************************************************************************

(defun mov_en (/)

  (noecho)
  (r_non)

  (setq p (getpoint "\nSeleccione el punto de aplicación del axil "))
  (if (/= p nil)
    (progn
      (setq ez (car p)
            ey (cadr p)
            Mz (* N ey)
            My (* N ez)
      )
      (if (or (= p3 nil) (= p4 nil))
        (zonaEN)
      )
      (ejen)
      (mov_en)
    )
```

```
  )
)

(defun C:MEN (/)

  (setq N 1000)
  (scp_cdg)
  (mov_en)
)

;*****************************************************************************
; Mensaje de saludo
;*****************************************************************************

(textpage)
(prompt "\nEste es un programa educativo.")
(prompt "\n\nUsted no esta autorizado para utilizarlo con cualesquiera otros
fines.\n")
(prompt "\n\n")
(setq ok "Copyright   A. Perez Garcia.   VERA CAAD I&D.    U.P.V.")
```

6.5 Módulo de aplicación de Vínculos con el contorno y descripción de Ligaduras entre barras

Los vínculos de la estructura al contorno pueden describirse utilizando elementos constructivos (zapatas centradas, zapatas de medianera, zapatas de esquina, riostras, etc.) o bien establecer la conexión mediante objetos ideales (articulación, apoyo deslizante, empotramiento, etc.).

6.5.1 Vínculos de tipo constructivo

Las siguientes rutinas permiten introducir y gestionar entidades de tipo BLOQUE que representan elementos constructivos de cimentación.

```
; ************ FUNCIONES PARA GENERAR ELEMENTOS DE CIMENTACION  ****************

; zco          INSERTA BLOQUES DE ZAPATAS COMBINADAS:  ZCOMB o ZCOMBE
; inapoyp      INSERTA  APOYOS GENERICOS PLANOS
; inapoye      INSERTA  APOYOS GENERICOS ESPACIALES
; C:zcen       INSERTA EL BLOQUE ASOCIADO A UNA ZAPATA CENTRADA
; C:zmed       INSERTA EL BLOQUE ASOCIADO A UNA ZAPATA DE MEDIANERA
; C:zesq       INSERTA EL BLOQUE ASOCIADO A UNA ZAPATA DE ESQUINA
; rio          FUNCION PARA INSERTAR  VIGAS RIOSTRAS  BLOQUE VRIOS
; C:RIO
; abalas       RE-ASIGNACION DEL BLOQUE BALASTO UN ELEMENTO O GRUPO ELEMENTOS
;              (BORRA SI EXISTE)
```

```
; bormoll      BORRA LOS BLOQUES BALASTO QUE EXISTAN EN UN PUNTO
; abas         ASIGNACION BLOQUE BALASTO SIN BORRAR LOS QUE EXISTAN


(prompt "Cargando las utilidades de CIMENTACION \n")


;*****************************************************************************
;* * *  INSERTA BLOQUES DE ZAPATAS COMBINADAS:  ZCOMB o ZCOMBE
;*****************************************************************************
'
(defun zco ( tt / p1 p2 p3 po d1)

   (noecho)
   (scpu)
   (cposm)
   (r_fin)
   (setq p1 (getpoint "\nBase Primer Soporte:"))
   (setq p2 (getpoint p1 "\nBase Segundo Soporte:"))
   (r_non)
   (setq p3  (trans p1 1 0))
   (VectorZ p1 p2)
   (setq po  (trans p3 0 1))
   (setq d1 (distance p1 p2))
   (if (equal tt 1) (command "_INSERT" "ZCOMB" po "XYZ" 1.3 0.6 d1 0))
   (if (equal tt 2) (command "_INSERT" "ZCOMBE" po "XYZ" 1.3 0.6 d1 0))
   (if (equal tt 5) (command "_INSERT" "ZCOMBB" po "XYZ" 1.3 0.6 d1 0))
   (if (equal tt 3) (command "_INSERT" "ZCORR" po "XYZ" 1.2 1 d1 0))
   (if (equal tt 4) (command "_INSERT" "ZCOBR" po "XYZ" 1.2 1 d1 0))
   (scpu)
   (pgosm)
)


;*****************************************************************************
;* * *   INSERTA  APOYOS GENERICOS PLANOS
;*****************************************************************************
'
(defun inapoyp ( tt / p1 osm)

   (noecho)
   (cposm)
   (r_fin)
   (setq p1 (getpoint "\n Base Soporte:"))
   (r_non)
   (if (equal tt 0) (command "_INSERT" "apoypg" p1 "" "" ""))
   (if (equal tt 1) (command "_INSERT" "apoyp1" p1 "" "" ""))
   (if (equal tt 2) (command "_INSERT" "apoyp2" p1 "" "" ""))
   (if (equal tt 3) (command "_INSERT" "apoyp3" p1 "" "" ""))
   (if (equal tt 4) (command "_INSERT" "apoyp4" p1 "" "" ""))
   (if (equal tt "5i") (command "_INSERT" "apoyp5i" p1 "" "" ""))
   (if (equal tt "5d") (command "_INSERT" "apoyp5d" p1 "" "" ""))
   (if (equal tt "6i") (command "_INSERT" "apoyp6i" p1 "" "" ""))
   (if (equal tt "6d") (command "_INSERT" "apoyp6d" p1 "" "" ""))
   (if (equal tt 7) (command "_INSERT" "apoyp7" p1 "" "" ""))
   (if (equal tt 8) (command "_INSERT" "apoyp8" p1 "" "" ""))
   (if (equal tt 9) (command "_INSERT" "apoyp9" p1 "" "" ""))
   (if (equal tt 10) (command "_INSERT" "apoyp10" p1 "" "" ""))
   (if (equal tt 11) (command "_INSERT" "apoyp11" p1 "" "" ""))
   (if (equal tt 12) (command "_INSERT" "movfp" p1 "" "" ""))
   (pgosm)
)


;*****************************************************************************
;* * *  INSERTA  APOYOS GENERICOS ESPACIALES
```

```
;*****************************************************************************

(defun inapoye ( tt / p1 osm)

   (noecho)
   (cposm)
   (scpu)
   (r_fin)
   (setq p1 (getpoint "\n Base Soporte:"))
   (r_non)
   (if (equal tt 0) (command "_INSERT" "apoyeg" p1 "" "" ""))
   (if (equal tt 1) (command "_INSERT" "apoye1" p1 "" "" ""))
   (if (equal tt 2) (command "_INSERT" "apoye2" p1 "" "" ""))
   (if (equal tt 3) (command "_INSERT" "apoye3" p1 "" "" ""))
   (if (equal tt 4) (command "_INSERT" "apoye4" p1 "" "" ""))
   (if (equal tt 5) (command "_INSERT" "apoye5" p1 "" "" ""))
   (if (equal tt 6) (command "_INSERT" "apoye6" p1 "" "" ""))
   (if (equal tt 7) (command "_INSERT" "apoye7" p1 "" "" ""))
   (if (equal tt 8) (command "_INSERT" "apoye8" p1 "" "" ""))
   (if (equal tt 9) (command "_INSERT" "apoye9" p1 "" "" ""))
   (if (equal tt 10) (command "_INSERT" "apoye10" p1 "" "" ""))
   (if (equal tt 11) (command "_INSERT" "movfe" p1 "" "" ""))
   (pgosm)
)


;*****************************************************************************
;* * *   INSERTA EL BLOQUE ASOCIADO A UNA ZAPATA CENTRADA
;*****************************************************************************

(defun C:zcen ( / p1 osm)

   (noecho)
   (cposm)
   (r_fin)
   (setq p1 (getpoint "\nBase del Soporte:"))
   (r_non)
   (command "_INSERT" "zapc" "_SC" 1 p1 0)
   (pgosm)
)


;*****************************************************************************
;* * *   INSERTA EL BLOQUE ASOCIADO A UNA ZAPATA DE MEDIANERA
;*****************************************************************************

(defun C:zmed ( / p1 osm)

   (noecho)
   (cposm)
   (r_fin)
   (setq p1 (getpoint "\nBase del Soporte:"))
   (r_non)
   (command "_INSERT" "zapb1" "_SC" 1 p1)
   (pgosm)
)



;*****************************************************************************
;* * *   INSERTA EL BLOQUE ASOCIADO A UNA ZAPATA DE ESQUINA
;*****************************************************************************

(defun C:zesq ( / p1 osm)

   (noecho)
```

```
   (cposm)
   (r_fin)
   (setq p1 (getpoint "\nBase del Soporte:"))
   (r_non)
   (command "_INSERT" "zape1" "_SC" 1 p1)
   (pgosm)
)


;*****************************************************************************
;* * *  FUNCION PARA INSERTAR  VIGAS RIOSTRAS  BLOQUE VRIOS
;*****************************************************************************

(defun rio ( / p1 p2 p3 po d1 p b h bb hh osm)

   (noecho)
   (cposm)
   (cpscp)
   (scpu)
   (r_fin)

   (setq b 40.0 h 40.0)
   (setq p1 (getpoint "\n Base Primer Soporte:"))
   (setq pp p1)
   (while (/= pp nil)
          (setq p2 (getpoint p1 "\nBase Siguiente Soporte:"))
          (setq pp p2)
          (if (/= pp nil)
              (progn (setq p3  (trans p1 1 0))
                     (VectorZ p1 p2)
                     (setq po  (trans p3 0 1))
                     (setq d1 (distance p1 p2))
                     (setq p b)
                     (princ "\n Ancho   (cm) <")(princ p)(princ "> ")(setq b
(getreal))
                     (if (eq (eval b) nil) (setq b p))
                     (setq p h)
                     (princ "\n Canto    (cm) <")(princ p)(princ "> ")(setq h
(getreal))
                     (if (eq (eval h) nil) (setq h p))
                     (setq bb (/ b 100))
                     (setq hh (/ h 100))
                     (command "_INSERT" "VRIOS" po "XYZ" bb hh d1 0)
                     (scpu)
                     (setq p1 p2)
              )
          )
   )

   (pgosm)
   (pgscp)
)


(defun C:RIO () (rio))


;*****************************************************************************
;* * *  RE-ASIGNACION DEL BLOQUE BALASTO UN ELEMENTO O GRUPO ELEMENTOS
;       (BORRA SI EXISTE)
;*****************************************************************************

(defun abalas ( / conj p0 p1 p2 p3 p4 pt pins n ent v  bl r0 r1 r2 k30)

   (noecho)
   (diano)
```

```
   (princ "\n Coeficiente de BALASTO (Kp/cm3) <3.00>")(setq k30 (getreal))
   (if (eq (eval k30) nil) (setq k30 3.00))
   (setq pins (list 50 50 50) v 0)
   (command "_INSERT" "BALASTO" pins "" "" "" )
   (setq bl (entlast) r0 (entnext bl) r1 (entget r0))
   (setq r2 (cons 1 (rtos k30 2 2)))  (entmod (subst r2 (assoc 1 r1) r1))

   (command "_ATTDISP" "_ON")
   (while (not (setq conj (ssget))))
   (repeat (sslength conj)
           (setq ent (ssname conj v) n (entget ent))
           (if (= "3DFACE" (cdr (assoc 0 n)))
               (progn (setq p1 (cdr (assoc 10 n)) p2 (cdr (assoc 11 n))
                            p3 (cdr (assoc 12 n)) p4 (cdr (assoc 13 n))
                            pt (trans p1 0 1) )
                      (bormoll pt)
                      (command "_COPY" bl "" pins pt)
                      (setq pt (trans p2 0 1))  (bormoll pt)
                      (command "_COPY" bl "" pins pt)
                      (setq pt (trans p3 0 1))  (bormoll pt)
                      (command "_COPY" bl "" pins pt)
                      (setq pt (trans p4 0 1))  (bormoll pt)
                      (command "_COPY" bl "" pins pt)
               )
           )
           (setq v (+ v 1))
     )

   (entdel bl)
   (command "_ATTDISP" "Normal")

)


;*****************************************************************************
;* * *   BORRA LOS BLOQUES BALASTO QUE EXISTAN EN UN PUNTO
;*****************************************************************************

(defun bormoll ( pt /  vv c e n p1 p2)

   (setq p1 (mapcar '+ pt '(0.05 0.05 0.05))
         p2 (mapcar '- pt '(0.05 0.05 0.05))
         c  (ssget "_C" p1 p2)
         vv 0
   )
   (if (/= c nil)
       (repeat (sslength c)
               (setq e (ssname c vv) n (entget e))
               (if (= "INSERT" (cdr (assoc 0 n)))
                   (if (= "BALASTO" (cdr (assoc 2 n)))
                       (if (EQUAL pt (cdr (assoc 10 n)) 0.01) (entdel e)) ) )
               (setq vv (+ vv 1))
       )
    )
)


;*****************************************************************************
;* * *  ASIGNACION BLOQUE BALASTO SIN BORRAR LOS QUE EXISTAN
;*****************************************************************************

(defun abas ( / conj p0 p1 p2 p3 p4 pin n nv ent v vv bl r0 r1 r2 k30 lp ctr)

   (noecho)
   (diano)
   (setq v 0 )
```

```
(princ "\n Coeficiente de BALASTO (Kp/cm3) <3.00>")(setq k30 (getreal))
(if (eq (eval k30) nil) (setq k30 3.00))
(setq pin (list 50 50 50) p1(list 100 100 100))
(setq lp (list pin p1))
(command "_INSERT" "BALASTO" pin "" "" "" )
(setq bl (entlast))
(setq r0 (entnext bl)
      r1 (entget r0))
(setq r2 (cons 1 (rtos k30 2 2)))  (entmod (subst r2 (assoc 1 r1) r1))

(while (not (setq conj (ssget))))
(princ "\n Estoy calculando, espere por favor")
(repeat (sslength conj)
        (setq ent (ssname conj v) n (entget ent))
        (if (= "3DFACE" (cdr (assoc 0 n)))
            (progn (setq p1  (cdr (assoc 10 n))
                         p2  (cdr (assoc 11 n))
                         p3  (cdr (assoc 12 n))
                         p4  (cdr (assoc 13 n)))

                   (setq vv 0 vc 0 nv (length lp) ctr T)

                   (while ctr
                          (if (equal p1 (nth vv lp) 0.03) (setq vc 1))
                          (setq vv (+ vv 1))
                          (if (= vv nv) (setq ctr nil))
                          (if (= vc 1) (setq ctr nil))
                   )

                   (if (= vc 0) (setq lp (cons p1 lp)))
                   (setq vv 0 vc 0 nv (length lp) ctr T)

                   (while ctr
                          (if (equal p2 (nth vv lp) 0.03) (setq vc 1))
                          (setq vv (+ vv 1))
                          (if (= vv nv) (setq ctr nil))
                          (if (= vc 1) (setq ctr nil))
                   )

                   (if (= vc 0) (setq lp (cons p2 lp)))
                   (setq vv 0 vc 0 nv (length lp) ctr T)

                   (while ctr
                          (if (equal p3 (nth vv lp) 0.03) (setq vc 1))
                          (setq vv (+ vv 1))
                          (if (= vv nv) (setq ctr nil))
                          (if (= vc 1) (setq ctr nil))
                   )

                   (if (= vc 0) (setq lp (cons p3 lp)))
                   (setq vv 0 vc 0 nv (length lp) ctr T)

                   (while ctr
                          (if (equal p4 (nth vv lp) 0.03) (setq vc 1))
                          (setq vv (+ vv 1))
                          (if (= vv nv) (setq ctr nil))
                          (if (= vc 1) (setq ctr nil))

                   )

                    (if (= vc 0) (setq lp (cons p4 lp)))
             )
        )

        (setq v (+ v 1))
```

```
     )

     (setq lp (reverse lp) lp (cdr lp) lp (cdr lp) v 0)
     (repeat (length lp)

             (command "_COPY" bl "" pin (nth v lp))
             (setq v (+ v 1))

     )

   (entdel bl)

)
```

6.5.2 Vínculo de tipo ideal

Las rutinas que se transcriben a continuación permiten introducir en el modelo BLOQUES que representan vínculos ideales o bien relajar total o parcialmente las ligaduras entre algunos elementos.

```
; ************ FUNCIONES PARA GENERAR VINCULOS DE LA ESTRUCTURA CON EL CONTORNO
;              Y DEFINIR LAS LIGADURAS INTERNAS ENTRE BARRAS Y NUDOS


; iapoyh       INSERCION DE UN APOYO HABITUAL
; iapoye       INSERCION DE UN APOYO ESPECIAL
; chblok1      CAMBIO A BLOQUE DE APOYOS ESPECIALES
; chblok2      CAMBIO A BLOQUE DE APOYOS HABITUALES
;
; rotula       INSERCION DE UNA ROTULA PLANA O ESPACIAL
; C:RT         Idem a la función    rotula
; selpins      PUNTO DE INSERCION DE LOS BLOQUES: NUDOS  y  PATNUD
; carnud       DEFINICION DE LAS CARACTERISTICAS DEL NUDO
; asignud1     ASIGNACION DEL BLOQUE PATNUD A UNA SOLA BARRA
; actualpatn   ACTUALIZACION DE LOS VALORES DEL BLOQUE PATNUD
;
;


(prompt "\nCargando las utilidades de APOYOS \n")


;*****************************************************************************
;* * *  INSERCION DE UN APOYO HABITUAL
;*****************************************************************************

(defun iapoyh (aph / p) ; aph  Tipo de apoyo habitual

   (cposm)
   (if (= aph "OTROS") (chblok1)
       (progn (setvar "OSMODE" 1195)
              (setq p (getpoint "\n\nUbicaci¢n del vinculo seleccionado "))
              (r_non)
              (if (= aph "APOYP1")  (command "_INSERT" "apoyp1"  "esc" 1 p 0 ""))
```

```
             (if (= aph "APOYP2")  (command "_INSERT" "apoyp2"  "esc" 1 p 0 ""))
             (if (= aph "APOYP3")  (command "_INSERT" "apoyp3"  "esc" 1 p 0 ""))
             (if (= aph "APOYP4")  (command "_INSERT" "apoyp4"  "esc" 1 p 0 ""))
             (if (= aph "APOYPG")  (command "_INSERT" "apoypg"  "esc" 1 p 0 ""))
             (if (= aph "APOYP7")  (command "_INSERT" "apoyp7"  "esc" 1 p 0 ""))
             (if (= aph "APOE1")   (command "_INSERT" "apoye1"  "esc" 1 p 0 ""))
             (if (= aph "APOE2")   (command "_INSERT" "apoye2"  "esc" 1 p 0 ""))
             (if (= aph "APOE3")   (command "_INSERT" "apoye3"  "esc" 1 p 0 ""))
             (if (= aph "APOE4")   (command "_INSERT" "apoye4"  "esc" 1 p 0 ""))
             (if (= aph "APOEG")   (command "_INSERT" "apoyeg"  "esc" 1 p 0 ""))
             (if (= aph "BALASTO") (command "_INSERT" "balasto" "esc" 1 p 0 ""))
       )
   )
   (pgosm)
)

;*****************************************************************************
;* * *  INSERCION DE UN APOYO ESPECIAL
;*****************************************************************************

(defun iapoye (ape / p) ; apn  Tipo de apoyo habitual

   (cposm)
   (if (= ape "MAS") (chblok2)
      (progn (setvar "OSMODE" 1195)
             (setq p (getpoint "\n\nUbicaci¢n del vinculo seleccionado "))
             (r_non)
             (if (= ape "APOYP5D")  (command "_INSERT" "apoyp5d" "esc" 1 p 0 ""))
             (if (= ape "APOYP5I")  (command "_INSERT" "apoyp5i" "esc" 1 p 0 ""))
             (if (= ape "APOYP6D")  (command "_INSERT" "apoyp6d" "esc" 1 p 0 ""))
             (if (= ape "APOYP6I")  (command "_INSERT" "apoyp6i" "esc" 1 p 0 ""))
             (if (= ape "APOYP10")  (command "_INSERT" "apoyp10" "esc" 1 p 0 ""))
             (if (= ape "APOYP8")   (command "_INSERT" "apoyp8"  "esc" 1 p 0 ""))
             (if (= ape "APOE5")    (command "_INSERT" "apoye5"  "esc" 1 p 0 ""))
             (if (= ape "APOE6")    (command "_INSERT" "apoye6"  "esc" 1 p 0 ""))
             (if (= ape "APOE7")    (command "_INSERT" "apoye7"  "esc" 1 p 0 ""))
             (if (= ape "APOE8")    (command "_INSERT" "apoye8"  "esc" 1 p 0 ""))
             (if (= ape "APOE9")    (command "_INSERT" "apoye9"  "esc" 1 p 0 ""))
             (if (= ape "BALASTO")  (command "_INSERT" "balasto" "esc" 1 p 0 ""))
      )
   )
   (pgosm)
)


;*****************************************************************************
;* * *   CAMBIO A BLOQUE DE APOYOS ESPECIALES
;*****************************************************************************

(defun chblok1 (/ lyr pin)

  (cpcap)
  (if (not (wcmatch lyr "PANELES"))
   (command "_LAYER" "_T" "PANELES" "_ON" "PANELES" "_S" "PANELES" "")
  )
  (setq pin (list 0 0 0))
  (entdel panel)
  (command "_INSERT" "apoye" "esc" 1.5 pin "0.0")
  (pgcap)
)


;*****************************************************************************
;* * *   CAMBIO A BLOQUE DE APOYOS HABITUALES
;*****************************************************************************
```

```
(defun chblok2 (/ l pin)

  (cpcap)
  (if (not (wcmatch lyr "PANELES"))
    (command "_LAYER" "_T" "PANELES" "_ON" "PANELES" "_S" "PANELES" "")
  )
  (setq pin (list 0 0 0))
  (entdel panel)
  (command "_INSERT" "apoyh" "esc" 1.5 pin "0.0")
  (pgcap)
)


;*****************************************************************************
;* * * INSERCION DE UNA ROTULA PLANA O ESPACIAL
;*****************************************************************************

(defun rotula ( / pins)

   (->)
   (cposm)
   (r_fin)
   (setq pins (getpoint "Punto de inserción de la ROTULA "))
   (r_non)
   (command "_INSERT" "rotula" pins "" "" "")
   (pgosm)
)

(defun C:RT () (rotula))


;*****************************************************************************
;* * * SELECCION DEL PUNTO DE INSERCION DEL BLOQUE NUDOS
;*****************************************************************************

(defun selpins (/ a pp qq)

  (->)
  (cposm)
  (r_cer)
  (setq  pto (getpoint "\nSeleccione la barra ")
         ent (entget (ssname (ssget pto) 0))
         cap (cdr (assoc 8 ent))
         p   (cdr (assoc 10 ent))
         q   (cdr (assoc 11 ent))
         pto (trans pto 1 0)
         pp  (trans p 0 1)                ; Coordenadas del extremo inicial del
                                          ; eje en el SCP actual
         qq  (trans q 0 1)                ; Idem extremo final
  )
  (VectorZ pp qq)
  (pgosm)

  (setq  pto (trans pto 0 1)
         pp  (trans p 0 1)
         qq  (trans q 0 1)
         d1  (distance pto pp)
         d2  (distance pto qq)
  )

  (if (< d1 d2)
    (setq pins pp)
    (setq pins qq)
  )
)
```

```
;*****************************************************************************
;* * * DEFINICION DE LAS CARACTERISTICAS DEL NUDO
;*****************************************************************************
'
(defun carnud (nud /)                    ; nud        Tipo de Relajación del nudo

   (if (= nud "DX") (setq DX "Libre"))
   (if (= nud "DY") (setq DY "Libre"))
   (if (= nud "DZ") (setq DZ "Libre"))
   (if (= nud "GX") (setq GX "Libre"))
   (if (= nud "GY") (setq GY "Libre"))
   (if (= nud "GZ") (setq GZ "Libre"))
)


;*****************************************************************************
;* * * ASIGNACION DEL BLOQUE PATNUD A UNA SOLA BARRA
;*****************************************************************************
'
(defun asignud1 ( / )

   (cpcap)
   (cposm)
   (cpscp)

   (selpins)
   (command "_LAYER" "_S" cap "")
   (command "_INSERT" "nudos"  pins "" "" "")
   (command "_INSERT" "patnud" pto  "" "" "")
   (setq Dx "Fijo" Dy "Fijo" Dz "Fijo" Gx "Fijo" Gy "Fijo" Gz "Fijo" stp nil)
   (while (not stp) (C:SS))
   (actualpatn (entlast))
   (setq nu (ssget "X" (list (cons 2 "nudos"))))
   (command "_ERASE" nu "")

   (pgscp)
   (pgcap)
   (pgosm)

   (setq Dx "Fijo" Dy "Fijo" Dz "Fijo" Gx "Fijo" Gy "Fijo" Gz "Fijo")
)


;*****************************************************************************
;* * * ACTUALIZACION DE LOS VALORES DEL BLOQUE PATNUD
;*****************************************************************************
'
(defun actualpatn (ent / p p1 p2 p3 p4 p5 p6 n1 n2 n3 n4 n5 n6 )

   (setq n1 (entnext ent)
         n2 (entnext n1)
         n3 (entnext n2)
         n4 (entnext n3)
         n5 (entnext n4)
         n6 (entnext n5)
         p1 (entget n1)                          ; MOVIMIENTO X
         p2 (entget n2)                          ; MOVIMIENTO Y
         p3 (entget n3)                          ; MOVIMIENTO Z
         p4 (entget n4)                          ; GIRO X
         p5 (entget n5)                          ; GIRO Y
         p6 (entget n6))                         ; GIRO Z

   (setq p (cons 1 Dx))       (entmod (subst p (assoc 1 p1) p1))
   (setq p (cons 1 Dy))       (entmod (subst p (assoc 1 p2) p2))
   (setq p (cons 1 Dz))       (entmod (subst p (assoc 1 p3) p3))
```

```
   (setq p (cons 1 Gx))       (entmod (subst p (assoc 1 p4) p4))
   (setq p (cons 1 Gy))       (entmod (subst p (assoc 1 p5) p5))
   (setq p (cons 1 Gz))       (entmod (subst p (assoc 1 p6) p6))
)
```

6.6 Módulo de aplicación de Cargas

Las cargas pueden aplicarse sobre los elementos estructurales de dos formas:

- directamente en forma de fuerzas puntuales o distribuidas (lineal o superficialmente).
- indirectamente a través de entidades de tipo 3Dcara que representan paños de forjado.

6.6.1 Funciones para aplicar Cargas directamente

Este módulo está compuesto por las siguientes rutinas:

```
; ************ FUNCIONES PARA GENERAR LAS ENTIDADES QUE REPRESENTAN LAS CARGAS


; initcarg     INICIALIZACION DE LAS VARIABLES GENERALES DE CARGA
; pprop        MODIFICA LA HIPOTESIS A QUE SE ASOCIA EL PESO PROPIO
; planocar     ESTABLECE EL PLANO DE CARGAS
; dibfuer      DIBUJAR UNA FUERZA PUNTUAL EN EL SCP ACTUAL
; insfue       INSERCION DE UNA CARGA PUNTUAL CON CUALQUIER ORIENTACION
; acarpv       INSERTA UNA CARGA PUNTUAL VERTICAL
; dibmom       INSERCION DEL BLOQUE CORRESPONDIENTE A UN MOMENTO
; puntoaplic   ESTABLECE EL EXTREMO DE UNA BARRA Y LA DISTANCIA AL MISMO PARA
;              INICIAR LA APLICACION DE UNA CARGA UNIFORME O TRAPEZOIDAL
; dibcu        DIBUJAR UNA CARGA UNIFORME EN EL SCP ACTUAL
; inscuc       INSERCION DE UNA CARGA UNIFORME CONSTANTE CON CUALQUIER ORIENTACION
; inscut       INSERCION DE UNA CARGA UNIFORME TRAPEZOIDAL CON CUALQUIER ORIENTACION
; cv           COLOCA UNA CARGA UNIFORME VERTICAL ENTRE DOS PUNTOS
; cvb          COLOCA UNA CARGA UNIFORME VERTICAL SOBRE UNA O MAS BARRAS
; incarpla     INSERTAR UN BLOQUE CON LA CARGA UNIFORME APLICADA A UNA PLACA
; asigcarp     INSERCION DE UNA CARGA UNIFORME SOBRE ELEMENTO PLACA
; acarsu       ASIGNA CARGA SUPERFICIAL VERTICAL O PERPENDICULAR A UNA ZONA DEFINIDA
;              POR 4 VERTICES UTILIZANDO UNA MALLA POLIGONAL
; acaruv       ASIGNA CARGA SUPERFICIAL UNIFORME Y VERTICAL A VARIOS ELEMENTOS
;              DE FORJADO UTILIZANDO EL BLOQUE CORRESPONDIENTE
; actescf      MODIFICA EL VALOR DEL FACTOR DE ESCALA DE LAS FUERZAS
; combacer     INSERCION DE LAS COMBINACIONES DE CARGA DEL ACERO EA-95
; combhor      INSERCION DE LAS COMBINACIONES DE CARGA DEL HORMIGON EHE
; inscblstp    INSERCION DEL BLOQUE DE CARGA  SISMICA,  P-DELTA  o  TEMPERATURA
; ?capa_hip    REUTILIZA, ACTIVA Y ESTABLECE COMO ACTUAL LA CAPA   HIP01
```

```
(prompt "Cargando las utilidades de CARGAS \n")


;*****************************************************************************
;
; INICIALIZACIONES RELATIVAS A LA APLICACION DE CARGA
;*****************************************************************************
;

;*****************************************************************************
;
;VARIABLES DE AMBITO GENERAL
;*****************************************************************************
;

;  hip      Hip¢tesis activa
;  ef       Escala de fuerzas
;  tc       Tipo de carga
;  mcp      M¢dulo por defecto de la carga puntual
;  mcu      M¢dulo por defecto de la carga uniforme constante
;  mci      M¢dulo por defecto de la carga trapezoidal en extremo inicial
;  mcf      M¢dulo por defecto de la carga trapezoidal en extremo final
;  mce      M¢dulo por defecto de la carga uniforme sobre elemento placa

;*****************************************************************************
;

(if (= ef  nil) (setq ef   1.0 ))
(if (= tc  nil) (setq tcp "90" ))
(if (= mcp nil) (setq mcp  1   ))
(if (= mcu nil) (setq mcu  1   ))
(if (= mci nil) (setq mci  1   ))
(if (= mcf nil) (setq mcf  1   ))
(if (= mce nil) (setq mce  100 ))



;*****************************************************************************
;
;* * * MODIFICA LA HIPOTESIS A QUE SE ASOCIA EL PESO PROPIO
;*****************************************************************************
;

(defun pprop ()

  (setq  hpp (getreal "\nHipotesis a que se asocia el peso propio:")
         fpp (getreal "\nFactor del peso propio                 :")
  )
  (modest)
)


;*****************************************************************************
;
;* * * SELECCIONAR EL PLANO DE CARGA E INSERTAR EL PANEL DE CARGA ESPECIFICADO
;*****************************************************************************
;

(defun planocar (p / o x y i osm lyr)

   (cposm)
   (cpscp)
   (cpcap)
   (r_fmi)

   (setq o (getpoint "\nPunto inicial del vector OX ")
         x (getpoint "\nPunto final   del vector OX ")
         y (getpoint "\nPunto final   del vector OY ")
   )
   (r_non)
   (diano)
   (command "_UCS" "3P" o x y)
   (setq i (getpoint "\nPunto de insercion del panel "))
   (command "_LAYER" "_T" "PANELES" "_ON" "PANELES" "_S" "PANELES" "")
```

```
   (if (= p "CARPN")
       (command "_INSERT" p "_SC" 1.5 i "0.0" "" "")
       (command "_INSERT" p "_SC" 1.5 i "0.0")
   )

   (pgscp)
   (pgosm)
   (pgcap)
)

;*****************************************************************************
;* * * DIBUJAR UNA FUERZA PUNTUAL EN EL SCP ACTUAL
;*****************************************************************************

(defun dibfuer (p                          ; Punto de aplicaci¢n de la fuerza
                m                          ; Módulo
                o                          ; Orientación
                / e r s)

  (setq  r (polar p o 0.1)
         s (polar p o (* ef m))
  )
  (command "_PLINE" p "_W" 0 0.05 r "_W" 0 0 s "")
  (setq e s)
)


;*****************************************************************************
;* * * INSERCION DE UNA CARGA PUNTUAL
;*****************************************************************************

(defun insfue (tip                         ; Tipo y orientacion de la carga puntual
               p m / o r s)

  (if (/= nil m)
    (setq mcp m)
  )
  (cpcap)
  (?capa_hip)
  (if (= tip "ALFA")
    (progn (setq o (getorient p "\nOrientacion de la fuerza  "))
           (dibfuer p mcp o)
    )
    (if (= tip "MOMENTO")
      (dibmom p mcp)
      (progn (setq o (atof tip))
             (if (= nil (member o (list 0 90 180 270)))
               (prompt "\n\nLa orientaci¢n de la fuerza no es valida ")
               (progn (setq o (cvunit o "grado" "radian"))
                      (dibfuer p mcp o)
               )
             )
      )
    )
  )
  (pgcap)
)


;*****************************************************************************
;* * * CARGA PUNTUAL VERTICAL
;*****************************************************************************

(defun acarpv (/ p d m osm lyr)

  (cpscp)
```

```
  (cpcap)
  (cposm)

  (command "_UCS" "")
  (command "_UCS" "X" "90" "")
  (setq d 1.570796)
  (?capa_hip)
  (r_fmi)
  (setq p (getpoint "\nPunto de aplicacion de la Carga"))
  (if (= p nil)
    (setq p (puntoaplic))
  )
  (setq m (getreal "\nModulo de la fuerza (Ton )  "))
  (r_non)
  (dibfuer p m d)

  (pgscp)
  (pgcap)
  (pgosm)

)


;*****************************************************************************
;* * * INSERCION DEL BLOQUE CORRESPONDIENTE A UN MOMENTO
;*****************************************************************************

(defun dibmom (p                  ; Modulo del momento
               m                  ; Punto de aplicacion
               / mom ex  ey)

  (setvar "ATTDIA" 0)
  (setq  ex (* m ef)
         ey (if (< m 0)
              (* -1 m ef)
              (* m ef)
            )
  )
  (setq mom (rtos m 2 3))
  (command "_INSERT" "momento" p ex ey "0.0" mom)
  (setvar "ATTDIA" 1)
)


;*****************************************************************************
;* * * ESTABLECE EL EXTREMO DE UNA BARRA Y LA DISTANCIA AL MISMO PARA INICIAR
;       LA APLICACION DE UNA CARGA UNIFORME O TRAPEZOIDAL
;*****************************************************************************

(defun puntoaplic (/ ent eje ini fin lado d1 d2 long d p)

  (setq  ent  (entsel
                "Seleccione el lado de la barra donde se aplicara la fuerza \n"
              )
         eje  (entget (car ent))
         ini  (trans (cdr (assoc 10 eje)) 0 1)
         fin  (trans (cdr (assoc 11 eje)) 0 1)
         lado (cadr ent)
         d1   (distance ini lado)
         d2   (distance fin lado)
         long (distance ini fin)
         d    (getreal "Distancia desde el extremo de la barra:  ")
  )
  (if (> d1 d2)

    (setq d (- long d))
```

```
  )
  (setq p (puntint ini fin (/ d long)))
)


;*****************************************************************************
;* * * DIBUJAR UNA CARGA UNIFORME EN EL SCP ACTUAL
;*****************************************************************************

(defun dibcu (p                         ; Punto inicial aplicaci¢n de la carga
              q                         ; Punto final
              mi                ; Módulo inicial
              mf                ; Módulo final
              o                         ; Orientación
              / l i j u)

  (setq  i (dibfuer p mi o)
         l (ssget ult_ent)
  )
  (setq j (dibfuer q mf o))
  (ssadd (entlast) l)
  (command "_PLINE" i j "")
  (setq u (entlast))
  (ssadd u l)
  (command "_PEDIT" u "_J" l "" "")

)

; Versión depurada elaborada por Danial Carvajal

(DEFUN dibcu (p                         ; Punto inicial aplicaci¢n de la carga
              q                         ; Punto final
              mi                ; Módulo inicial
              mf                ; Módulo final
              o                         ; Orientación
              / l i j)

  (SETQ l (SSADD))
  (SETQ i (dibfuer p mi o))
  (SSADD (ENTLAST) l)
  (SETQ j (dibfuer q mf o))
  (SSADD (ENTLAST) l)
  (COMMAND "_PLINE" i j "")
  (SSADD (ENTLAST) l)
  (COMMAND "_PEDIT" (ENTLAST) "_J" l "" "")
)


;*****************************************************************************
;* * * INSERCION DE UNA CARGA UNIFORME CONSTANTE
;*****************************************************************************

(defun inscuc (o                 ;Tipo y orientaci¢n de la carga uniforme
               / mm p q  r s)

  (cposm)
  (r_fmi)
  (setq p (getpoint "\nExtremo INICIAL de la carga uniforme "))
  (if (= p nil)
    (setq p (puntoaplic))
  )
  (setq q (getpoint "\nExtremo FINAL   de la carga uniforme "))
  (if (= q nil)
    (setq q (puntoaplic))
  )
```

```
  (r_non)
  (setq mm (getreal "\nModulo de la carga (Toneladas/metro)  "))
  (if (/= nil mm)
    (setq mcu mm)
  )
  (?capa_hip)
  (if (= o "BETA")
    (progn (setq o (getorient p "\nOrientacion de la fuerza  "))
           (dibcu p q mcu mcu o)
    )
    (progn (setq o (- (atof o) 10))
           (if (= nil (member o (list 0 90 180 270)))
             (prompt "\n\nLa orientación de la fuerza no es valida ")
             (progn (setq o (cvunit o "grado" "radian"))
                    (dibcu p q mcu mcu o)
             )
           )
    )
  )
  (pgosm)
)


;*****************************************************************************
;* * * INSERCION DE UNA CARGA UNIFORME TRAPEZOIDAL
;*****************************************************************************

(defun inscut (o                  ; Tipo y orientaci¢n de la carga trapezoidal
               / p q mi  mf r s)

  (cposm)
  (r_fmi)
  (setq p (getpoint "\nExtremo INICIAL de la carga uniforme "))
  (if (= p nil)
    (setq p (puntoaplic))
  )
  (setq  mi (getreal
              "\nValor de la carga en este extremo (Toneladas/metro)  "
            )
  )
  (setq q (getpoint "\nExtremo FINAL   de la carga uniforme "))
  (if (= q nil)
    (setq q (puntoaplic))
  )
  (setq  mf (getreal
              "\nValor de la carga en este extremo (Toneladas/metro)  "
            )
  )
  (r_non)
  (if (/= nil mi)
    (setq mci mi)
  )
  (if (/= nil mf)
    (setq mcf mf)
  )
  (?capa_hip)
  (if (= o "BETA")
    (progn (setq o (getorient p "\nOrientacion de la fuerza  "))
           (dibcu p q mci mcf o)
    )
    (progn (setq o (- (atof o) 20))
           (if (= nil (member o (list 0 90 180 270)))
             (prompt "\n\nLa orientaci¢n de la fuerza no es valida ")
             (progn (setq o (cvunit o "grado" "radian"))
                    (dibcu p q mci mcf o)
             )
```

```
          )
    )
  )
  (pgosm)
)

;*********************************************************************
;* * *     COLOCA UNA CARGA UNIF. VERTICAL ENTRE DOS PUNTOS
;*********************************************************************

(defun cv (/ p p1 q q1 r pin pj mm x1 y1 z1 x2 y2 z2 zz osm lyr)

  (cposm)
  (cpcap)
  (cpscp)
  (scpu)
  (r_fmi)

  (setq p (getpoint "\nExtremo INICIAL de la carga uniforme"))
  (if (= p nil)
    (progn
      (pgosm)
      (pgscp)
      (cvb)
    )
    (progn
      (setq p1 p)
      (setq q 1)
      (while (/= q nil)
        (r_fmi)
        (setq q (getpoint p1 "\nExtremo FINAL   de la carga uniforme"))
        (r_non)
        (if (/= q nil)
          (progn
            (setq x1 (car p)
                  y1 (cadr p)
                  z1 (caddr p)
                  x2 (car q)
                  y2 (cadr q)
                  z2 (caddr q)
                  q1 (list x2 y2 z1)
                  z1 (+ 100 z1)
                  r  (list x1 y1 z1)
            )
            (command "_UCS" "_3p" p q1 r)
            (setq mm (getreal "\nModulo de la carga (Toneladas/metro)"))
            (if (/= nil mm)
              (setq mcu mm)
            )
            (if (= 0 mcu)
              (setq mcu 1)
            )
            (?capa_hip)
            (setq pin (trans p 0 1)
                  pj  (trans q 0 1)
            )
            (dibcu pin pj mcu mcu 1.5708)
            (scpu)
            (setq p  q
                  p1 q
            )
          )
        )
      )
    )
  )
```

```
  )
  (pgcap)
  (pgscp)
  (pgosm)
)


(defun C:CV () (cv))


;*********************************************************************
;* * *     COLOCA UNA CARGA UNIF. VERTICAL SOBRE VARIAS BARRAS
;*********************************************************************

(defun cvb (/ p p1 q q1 r pin pj mm x1 y1 z1 x2 y2 zz conj v n ent osm)

  (cposm)
  (r_non)
  (prompt "Seleccione las barras a cargar")
  (while (not (setq conj (ssget))))

  (if (/= conj nil)
    (progn (cpscp)
           (scpu)
           (cpcap)
           (?capa_hip)
           (setq mm (getreal "\nModulo de la carga (Toneladas/metro)"))
           (if (/= nil mm)
             (setq mcu mm)
           )
           (if (= 0 mcu)
             (setq mcu 1)
           )
           (setq v 0)
           (repeat (sslength conj)
             (setq ent (ssname conj v)
                   n   (entget ent)
             )
             (if (= "LINE" (cdr (assoc 0 n)))
               (progn (setq p  (cdr (assoc 10 n))
                            q  (cdr (assoc 11 n))
                            x1 (car p)
                            y1 (cadr p)
                            z1 (caddr p)
                            x2 (car q)
                            y2 (cadr q)
                            z2 (caddr q)

                            q1 (list x2 y2 z1)
                            z1 (+ 100 z1)
                            r  (list x1 y1 z1)
                      )
                      (if (and (= x1 x2) (= y1 y2))
                        (setq v (+ v 1))
                        (progn (command "_UCS" "_3p" p q1 r)
                               (setq pin (trans p 0 1)
                                     pj  (trans q 0 1)
                               )
                               (dibcu pin pj mcu mcu 1.5708)
                               (setq v (+ v 1))
                               (scpu)
                        )
                      )
               )
             )
           )
    )
```

```
          (pgscp)
          (pgcap)
    )
  )
  (pgosm)
)


(defun C:CVB () (cvb))


;*****************************************************************************
;* * * INSERTAR UN BLOQUE CON LA CARGA UNIFORME APLICADA A UNA PLACA
;*****************************************************************************

(defun incarpla  (ent pt / rot v p poi p1 p2 nomp1 nomp2)

  (noecho)
  (cposm)
  (r_non)
  (if (= escarpn nil)
    (setq escarpn 1.0)
  )
  (command "_INSERT" "CARPN" "_SC" escarpn pt "")
  (setq  rot   (cdr (assoc 5 (entget ent)))
         nomp1 (entnext (entlast))
         nomp2 (entnext nomp1)
         p1    (entget nomp1)
         p2    (entget nomp2)
         p     (cons 1 " ")
  )
  (entmod (subst p (assoc 1 p1) p1))
  (setq  v (rtos mce 2 1)
         p (cons 1 v)
  )
  (entmod (subst p (assoc 1 p2) p2))
  (pgosm)
)


;*****************************************************************************
;* * * INSERCION DE UNA CARGA UNIFORME SOBRE ELEMENTO PLACA
;*****************************************************************************

(defun asigcarp  (/ cp conj mi mf p0 p1 p2 p3 p4 p5 pt n0 n2 v ent)

  (noecho)
  (diano)
  (cposm)
  (r_non)
  (cpcap)

  (setq  cp (getreal
              "\nModulo de la carga aplicada sobre la placa (Kp/m2):"
            )
  )
  (if (/= nil cp)
    (setq mce cp)
  )
  (?capa_hip)
  (while (not (setq conj (ssget))))
  (setq v 0)
  (repeat (sslength conj)
    (setq ent (ssname conj v)
          v   (+ v 1)
          n2  (entget ent)
```

```
    )
    (if (= "3DFACE" (cdr (assoc 0 n2)))
      (progn (setq p1 (cdr (assoc 10 n2))
                   p2 (cdr (assoc 11 n2))
                   p3 (cdr (assoc 12 n2))
                   p4 (cdr (assoc 13 n2))
                   p0 (pmig p1 p2)
                   p5 (pmig p3 p4)
                   pt (trans (pmig p0 p5) 0 1)
             )
             (incarpla ent pt)
      )
    )
  )
  (pgcap)
  (diasi)
  (pgosm)

)


;*****************************************************************************
;* * * CARGA SUPERFICIAL VERTICAL DEFINIDA POR UNA AREA DE 4 PUNTOS
;*****************************************************************************
(defun acarsu (tip  /     p1   p2   p3    p4   p5   p6   p7   p8    cp
               fc   x     y    z    ent   n    tl   tll  cap  entac
               fich osm   lyr
              )

  (cpscp)
  (cposm)
  (cpcap)
  (scpu)
  (r_fmi)

  (setq p1 (getpoint "\n Punto inicial primera esquina"))

  (setq p2 (getpoint p1 "\n Punto 2ª esquina"))
  (setq p3 (getpoint p2 "\n Punto 3ª esquina"))
  (setq p4 (getpoint p3 "\n Punto 4ª esquina"))
  (setq
    cp (getreal
         "\n Modulo de la carga aplicada sobre la superf. (Kp/m2):"
       )
  )
  (if (/= nil cp)
    (setq mce cp)
  )
  (setq fc (* ef (/ mce 1000)))
  (if (= tip 2)
    (progn
      (command "_UCS" "_3p" p1 p2 p3)
      (setq p1 (trans p1 0 1)
            p2 (trans p2 0 1)
            p3 (trans p3 0 1)
            p4 (trans p4 0 1)
      )
    )
  )
  (setq  x (nth 0 p1)
         y (nth 1 p1)
         z (nth 2 p1)
  )
  (setq  z  (+ z fc)
         p5 (list x y z)
```

```
  )
  (setq  x (nth 0 p2)
         y (nth 1 p2)
         z (nth 2 p2)
  )
  (setq  z  (+ z fc)
         p6 (list x y z)
  )
  (setq  x (nth 0 p3)
         y (nth 1 p3)
         z (nth 2 p3)
  )
  (setq  z  (+ z fc)
         p7 (list x y z)
  )
  (setq  x (nth 0 p4)
         y (nth 1 p4)
         z (nth 2 p4)
  )
  (setq  z  (+ z fc)
         p8 (list x y z)
  )
  (r_non)
  (?capa_hip)
  (pbase p1 p2 p3 p4 p5 p6 p7 p8)
  (setq ent (entlast))
  (setq  tl  "CARGASUP"
         tll (cons 6 tl)
  )
  (if (= nil (tblsearch "LTYPE" tl))
    (progn (CreaTl tl)
           (CargaTl tl)
    )
  )
  (setq fich (open "c:/cid/cad/st.lin" "w"))
  (close fich)
  (setq n (entget ent))
  (setq cap (assoc 6 n))
  (if (= nil cap)
    (setq entac (cons tll n))
    (setq entac (subst tll cap n))
  )
  (entmod entac)

  (pgscp)
)


;*****************************************************************************
;* * * CARGA SUPERFICIAL UNIFORME VERTICAL
;*****************************************************************************

(defun acaruv ( / )

    (cpscp)
    (scpu)
    (asigcarp)
    (pgscp)
)


;*****************************************************************************
;* * * MODIFICA EL VALOR DEL FACTOR DE ESCALA DE LAS FUERZAS
;*****************************************************************************

(defun actescf (/ conj p n1 n2 n3 esc lyr)
```

```
  (setq ef   (getreal "\nFactor de escala de cargas:")
        conj (ssget "X" (list (cons 2 "TIPEST")))
  )
  (if (/= nil conj)
    (if (> (sslength conj) 1)
      (prompt "Elimine los bloque TIPEST sobrantes ")
      (progn (setq n1  (ssname conj 0)
                   n2  (entnext n1)
                   n3  (entnext n2)
                   esc (entget n3)
                   p   (cons 1 (rtos ef 2 2))
             )
             (entmod (subst p (assoc 1 esc) esc))
             (entupd n1)
      )
    )
    (progn (cpcap)
           (setq p (getpoint
                     "\nPunto de inserci¢n del bloque TIPO DE ESTRUCTURA "
                   )
           )
           (command "_LAYER"     "_T"         "TIPOMATS"   "_ON"
                    "TIPOMATS"   "_S"         "TIPOMATS"   ""
                   )
           (command "_INSERT" "TIPEST" "_SC" 3 p "0.0")
           (pgcap)
    )
  )
)


;*****************************************************************************
;* * * INSERCION DE LAS COMBINACIONES DE CARGA DEL HORMIGON EHE
;*****************************************************************************

(defun combhor (/ p ent)

  (setq p (getpoint
            "\nPunto de inserci¢n del bloque COMBINACIONES DE CARGA"
          )
  )
  (command "_INSERT" "COMBHORM" "_SC" 1 p "0.0")
  (setq ent (entlast))
  (command "_EXPLODE" ent)
)



;*****************************************************************************
;* * * INSERCION DE LAS COMBINACIONES DE CARGA DEL ACERO EA-95
;*****************************************************************************

(defun combacer  (/ p ent)

  (setq p (getpoint
            "\nPunto de inserci¢n del bloque COMBINACIONES DE CARGA"
          )
  )
  (command "_INSERT" "COMBACER" "_SC" 1 p "0.0")
  (setq ent (entlast))
  (command "_EXPLODE" ent)
)


;*****************************************************************************
```

```
;* * * INSERCION DEL BLOQUE DE CARGA  SISMICA,  P-DELTA  o  TEMPERATURA
;*****************************************************************************

(defun inscblstp (tp                      ; tp=1 Sismo
                                          ; tp=2 P-Delta
                                          ; tp=3 temperatura
                  / l p )

  (command "_UCS" "U")
  (EjeZ)
  (setq l (getvar "CLAYER"))
  (if (not (wcmatch l "COMBINA_HIPOT"))
    (command "_LAYER" "_S" "COMBINA_HIPOT" "")
  )
  (setq  p (getpoint
             (strcat "\n Punto de inserci¢n del BLOQUE de carga ")
           )
  )
  (if (= tp 1)
    (command "_INSERT" "BLSISM" p "" "" "")
  )
  (if (= tp 2)
    (command "_INSERT" "BLPDELTA" p "" "" "")
  )
  (if (= tp 3)
    (command "_INSERT" "CARTEMP" p "" "" "")
  )

)


;*****************************************************************************
;* * * REUTILIZA, ACTIVA Y ESTABLECE COMO ACTUAL LA CAPA   HIP01
;*****************************************************************************

(defun ?capa_hip ( / ll l)

   (setq ll (getvar "CLAYER"))
   (setq l (strcase ll nil))
   (if (not (wcmatch l "HIP*"))

       (command "_LAYER" "_T" hip "_ON" hip "_S" hip "")
   )
)
```

6.6.2 Funciones para aplicar Cargas a través de los forjados

```
; ************ FUNCIONES PARA APLICAR CARGAS A TRAVES DE ELEMENTOS TIPO FORJADO

; forjavo    INTRODUCION DE FORJADO VOLADIZO
; C:FV
; forjauni   INTRODUCION DE FORJADO UNIDIRECCIONAL
; C:FU
; forjabi    INTRODUCION DE FORJADO BIDIRECIONAL
; C:FB
; forjabis   INTRODUCCION DE UNA PORCION DE PAÑO DE FORJADO
```

```
(prompt "Cargando las utilidades de FORJADO \n")


;*****************************************************************************
;* * * INTRODUCION DE FORJADO VOLADIZO
;*****************************************************************************

(defun forjavo (/ p1 p2 p3 p4 osm)

  (cposm)
  (r_fmi)
  (setq p1 (getpoint "\nPrimer extremo  viga 1 "))
  (setq p2 (getpoint p1 "\nSegundo extremo viga 1 "))
  (setq p3 (getpoint p2 "\nPrimer extremo del voladizo "))
  (setq p4 (getpoint p3 "\nSegundo extremo del voladizo "))
  (forjabis p1 p2 p3 p4)
  (pgosm)

)

(defun C:FV () (forjavo))


;*****************************************************************************
;* * * INTRODUCION DE FORJADO UNIDIRECCIONAL
;*****************************************************************************

(defun forjauni  (/ p1 p2 p3 p4 p5 p6 osm)

  (cposm)
  (r_fmi)
  (setq  p1 (getpoint "\nPrimer extremo  viga 1:")
         p2 (getpoint p1 "\nSegundo extremo viga 1:")
         p3 (getpoint p2 "\nPrimer extremo viga  2:")
         p4 (getpoint p3 "\nSegundo extremo viga 2:")
         p5 (pmig p1 p4)
         p6 (pmig p2 p3)
  )
  (forjabis p1 p2 p6 p5)
  (forjabis p3 p4 p5 p6)
  (pgosm)
)

(defun C:FU () (forjauni))


;*****************************************************************************
;* * * INTRODUCION DE FORJADO BIDIRECIONAL
;*****************************************************************************

(defun forjabi (/ poi pj pk pl pij pjk pkl pil pc bi bj bk bl osm)

  (cposm)
  (r_fin)
  (setq  poi (getpoint "\nPrimer  vertice del recuadro:")
         pj  (getpoint "\nSegundo vertice del recuadro:")
         pk  (getpoint "\nTercer  vertice del recuadro:")
         pl  (getpoint "\nCuarto  vertice del recuadro:")
         pij (pmig poi pj)
         pjk (pmig pj pk)
         pkl (pmig pk pl)
         pli (pmig pl poi)
         a   (distance pli pjk)
         b   (distance pij pkl)
```

```
        a/b (/ a b)
  )
  (r_non)

  (setq tl "_FBID")
  (if (= nil (tblsearch "LTYPE" tl))
    (progn (CreaTl tl)
           (CargaTl tl)
    )
    (command "_LINETYPE" "_S" tl "")
  )

  (setq fich (open "c:/cid/cad/st.lin" "w"))
  (close fich)

  (if (and (<= a/b 1.2) (>= a/b 0.8))
    (progn (setq pc (pmig pij pkl))
           (forjabis poi pj pc pc)
           (forjabis pj pk pc pc)
           (forjabis pk pl pc pc)
           (forjabis pl poi pc pc)
    )
    (if (> a b)
      (progn (setq pcil (polar pli (angle pli pjk) (distance poi pli)))
             (setq pcjk (polar pjk (angle pjk pli) (distance pj pjk)))

             (forjabis poi pj pcjk pcil)
             (forjabis pj pk pcjk pcjk)
             (forjabis pk pl pcil pcjk)
             (forjabis pl poi pcil pcil)
      )
      (progn (setq pcij (polar pij (angle pij pkl) (distance poi pij)))
             (setq pckl (polar pkl (angle pkl pij) (distance pk pkl)))

             (forjabis poi pj pcij pcij)
             (forjabis pj pk pckl pcij)
             (forjabis pk pl pckl pckl)
             (forjabis pl poi pcij pckl)
      )
    )
  )
  (command "_LINETYPE" "_S" "" "")
  (pgosm)
)

(defun C:FB () (forjabi))



;*****************************************************************************
;* * * INTRODUCCION DE UNA PORCION DE PAÑO DE FORJADO
;*****************************************************************************

(defun forjabis  (p1 p2 p3 p4)

  (r_non)
  (3_CARA p1 p2 p3 p4)
)
```

6.7 Módulo de Cálculo y Trazado de Forjados

```
; **********  CALCULO DE ARMADURAS Y TRAZADO FORJADOS UNIDIRECCIONALES

; carac       PIDE LAS CARACTERISTICAS DEL FORJADO
; forj        Pide datos, los salva, llama al calculo, lee resultados
;             y llama a la funcion   ponneg   para dibujar los negativos
; c:forj      Idem a la función    forj
; salva       SALVA LOS DATOS DE FORJADO PARA UTILIZARLOS EN EL CALCULO
; neg         Dibuja manualmente un negativo
; c:neg       Idem a la función   neg
; dibneg      DIBUJA LAS ARMADURAS DE NEGATIVO
; dibtot      LLAMA AL FICHERO DE RESULTADOS Y LO DIBUJA TODO
; ponpos      FUNCION PARA COLOCAR LA ARMADURA DE POSITIVOS


(prompt "Cargando las utilidades de CALCULO Y DIBUJO ARMADO FORJADOS \n")



;*****************************************************************************
;* * * INICIALIZACIONES
;*****************************************************************************

(setq g      50
      h      (* 0.002 g)
      a      (* 0.005 g)
      b      (* 0.0015 g)
      gf     (* 0.5 h)
      q      (* 0.0008 g)
      p      (* 0.5 q)
      escala (/ g 100)
)
(setvar "DIMASZ" gf)
(setvar "DIMEXE" b)
(setvar "DIMEXO" a)
(setvar "DIMTSZ" p)
(setvar "DIMDLE" q)
(setvar "DIMTIX" 0)
(setvar "DIMZIN" 0)
(setvar "LUNITS" 2)

(command "_STYLE" "cotas" "" h 1.0 0.0 "" "" "")

(cond ((>= g 50)
       (setvar "DIMLFAC" 1)
       (setvar "DIMRND" 0.01)
       (setvar "LUPREC" 2)
      )
      ((and (< g 50) (> g 5))
       (setvar "DIMLFAC" 100)
       (setvar "DIMRND" 0.1)
       (setvar "LUPREC" 1)
      )
      ((<= g 5)
       (setvar "DIMLFAC" 1000)
       (setvar "DIMRND" 1)
       (setvar "LUPREC" 0)
      )
)

(if (= nil (tblsearch "LTYPE" "VOL_SEC"))
```

```
  (progn (CreaTl "VOL_SEC")
         (CargaTl "VOL_SEC")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI4"))
  (progn (CreaTl "_FI4")
         (CargaTl "_FI4")
  )
)

(if (= nil (tblsearch "LTYPE" "_FI5"))
  (progn (CreaTl "_FI5")
         (CargaTl "_FI5")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI6"))
  (progn (CreaTl "_FI6")
         (CargaTl "_FI6")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI8"))
  (progn (CreaTl "_FI8")
         (CargaTl "_FI8")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI10"))
  (progn (CreaTl "_FI10")
         (CargaTl "_FI10")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI12"))
  (progn (CreaTl "_FI12")
         (CargaTl "_FI12")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI14"))
  (progn (CreaTl "_FI14")
         (CargaTl "_FI14")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI16"))
  (progn (CreaTl "_FI16")
         (CargaTl "_FI16")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI20"))
  (progn (CreaTl "_FI20")
         (CargaTl "_FI20")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI25"))
  (progn (CreaTl "_FI25")
         (CargaTl "_FI25")
  )
)
(if (= nil (tblsearch "LTYPE" "_FI32"))
  (progn (CreaTl "_FI32")
         (CargaTl "_FI32")
  )
)

(setq fich (open "c:/cid/cad/st.lin" "w"))
(close fich)

(setq b      10
      h      30
```

```
      b0     70
      h0     5
      fck    25
      fyk    400
      gf     1.6
      sim    "SIMETRICA"
      tipfor "Viguetas Pretens"
      g       5.50
      q       2.00
      cp1    0.0
      a       0
      dmin   10
)
(command "_LAYER" "_N" "negat" "")
(command "_LAYER" "_N" "posit" "")
(command "_LAYER" "_N" "textarm" "")
(command "_LAYER" "_N" "kgnegat" "")
(scpu)
(command "_PLAN" "")


;*****************************************************************************
;* * *    PIDE LAS CARACTERISTICAS DEL FORJADO
;*****************************************************************************
'
(defun carac (/ bb)

  (noecho)
  (cpscp)
  (scpu)

  (setq bb fck)
  (princ "\n Hormigon  fck          (N/mm2) H<")
  (princ bb)
  (princ ">")
  (setq fck (getreal))
  (if (eq (eval fck) nil)
    (setq fck bb)
  )
  (setq bb fyk)
  (princ "\n Acero     fyk          (N/mm2) B<")
  (princ bb)
  (princ "> ")
  (setq fyk (getreal))
  (if (eq (eval fyk) nil)
    (setq fyk bb)
  )
  (setq bb gf)
  (princ "\n Coef. mayoracion      (gamma f) <")
  (princ bb)
  (princ "> ")
  (setq gf (getreal))
  (if (eq (eval gf) nil)
    (setq gf bb)
  )
  (setq bb b)
  (princ "\n Ancho Nervio               (cm) <")
  (princ bb)
  (princ "> ")
  (setq b (getreal))
  (if (eq (eval b) nil)
    (setq b bb)
  )
  (setq bb h)
  (princ "\n Canto total                (cm) <")
  (princ bb)
```

```
  (princ "> ")
  (setq h (getreal))
  (if (eq (eval h) nil)
    (setq h bb)
  )
  (setq bb h0)
  (princ "\n Canto losa comprs.(alas)   (cm) <")
  (princ bb)
  (princ "> ")
  (setq h0 (getreal))
  (if (eq (eval h0) nil)
    (setq h0 bb)
  )
  (setq bb b0)
  (princ "\n Intereje viguetas          (cm) <")
  (princ bb)
  (princ "> ")
  (setq b0 (getreal))
  (if (eq (eval b0) nil)
    (setq b0 bb)
  )
  (setq bb nil)
  (princ "\n Armadura  (S-imetrica//A-simetrica) <")
  (princ sim)
  (princ "> ")
  (setq bb (getstring))
  (setq bb (strcase bb nil))
  (if (= bb "S")
    (setq sim "SIMETRICA")
  )
  (if (= bb "A")
    (setq sim "ASIMETRICA")
  )
  (setq bb dmin)
  (princ "\n Diametro armadura minimo   (mm) <")
  (princ bb)
  (princ "> ")
  (setq dmin (getreal))
  (if (eq (eval dmin) nil)
    (setq dmin bb)
  )
  (setq bb nil)
  (princ
    "\n Tipo forjado  (Viguetas [P]retens//Viguetas [A]rmadas//Nervios [I]n situ)<"
  )
  (princ tipfor)
  (princ "> ")
  (setq bb (getstring))
  (setq bb (strcase bb nil))
  (if (= bb "P")
    (setq tipfor "Viguetas Pretens")
  )
  (if (= bb "A")
    (setq tipfor "Viguetas Armadas")
  )
  (if (= bb "I")
    (setq tipfor "In situ")
  )
  (pgscp)
)


;*****************************************************************************
;* * *   Pide datos, los salva, llama al calculo, lee resultados
;       y llama a la funcion   ponneg   para dibujar los negativos
;*****************************************************************************
```

```
(defun forj   (/    nv    p1   p2   p3    d1   pp    an   bb   lap   lluz
               lan  lg    lq   lp   la    exi  exd  bb   fckk fykk oma
              )

  (cposm)
  (noecho)
  (setq  fckk (* 10 fck)
         fykk (* 10 fyk)
  )
  (princ "\n fck=")
  (princ fck)
  (princ "  fyk=")
  (princ fyk)
  (princ " Nervio=")
  (princ b)
  (princ "  Canto=")
  (princ h)
  (princ "  Losa=")
  (princ h0)
  (princ "  Intereje=")
  (princ b0)
  (princ "    Armadura ")
  (princ sim)
  (princ "  Gamma f=")
  (princ gf)
  (scpu)
  (setq  nv   0
         bb   0.0
         p1   (getpoint "\nPrimer Apoyo:")
         lap  (list p1)
         lg   (list bb)
         lq   (list bb)
         lp   (list bb)
         la   (list bb)
         lluz (list bb)
         an   10.0
         bb   an
  )
  (princ "\n Ancho Apoyo              (cm) <")
  (princ bb)
  (princ ">")
  (setq an (getreal))
  (if (eq (eval an) nil)
    (setq an bb)
  )
  (setq  lan (list an)
         pp  1
  )

  (while (/= pp nil)
    (setq p2 (getpoint p1 "\nSiguiente Apoyo:")
          pp p2
    )
    (if (/= pp nil)
      (progn
        (setq nv (1+ nv)
              bb an
        )
        (princ "\n Ancho Apoyo              (cm) <")
        (princ bb)
        (princ "> ")
        (setq an (getreal))
        (if (eq (eval an) nil)
          (setq an bb)
        )
```

```
        (setq bb g)
        (princ "\n Carga Permanente       (KN/m2) <")
        (princ bb)
        (princ "> ")
        (setq g (getreal))
        (if (eq (eval g) nil)
          (setq g bb)
        )
        (setq bb q)
        (princ "\n Carga Variable         (KN/m2) <")
        (princ bb)
        (princ "> ")
        (setq q (getreal))
        (if (eq (eval q) nil)
          (setq q bb)
        )
        (setq bb cp1)
        (princ "\n Carga Puntual             (KN) <")
        (princ bb)
        (princ "> ")
        (setq cp1 (getreal))
        (if (eq (eval cp1) nil)
          (setq cp1 bb)
        )
        (if (> cp1 0.0)
          (progn
            (setq bb a)
            (princ "\n Posicion Carga Puntual a Ext.IZQ (m) <")
            (princ bb)
            (princ "> ")
            (setq p3 (getpoint p1 "\nMarque Posicion"))
            (setq a (distance p1 p3))
            (if (< a 0.15)
              (setq a 0)
            )
          )
        )
        (setq lap  (cons p2 lap)
              lan  (cons an lan)
              lg   (cons g lg)
              lq   (cons q lq)
              lp   (cons cp1 lp)
              la   (cons a la)
              d1   (distance p1 p2)
              lluz (cons d1 lluz)
              p1   p2
        )
      )
    )
  )
)
(princ
  "\n EXTREMO INICIAL   <V-oladizo//A-poyo//E-mpotr.//P-rolongación> "
)
(setq exi (getstring))
(princ
  "\n EXTREMO FINAL     <V-oladizo//A-poyo//E-mpotr.//P-rolongación> "
)
(setq exd (getstring))
(setq  exi  (strcase exi nil)
       exd  (strcase exd nil)
       lluz (reverse lluz)
       lg   (reverse lg)
       lq   (reverse lq)
       lap  (reverse lap)
       lan  (reverse lan)
       lp   (reverse lp)
```

```
        la   (reverse la)
  )
  (salva nv lluz lg lq lan lp la exi exd fckk fykk b h b0 h0 sim dmin)
  (r_non)
  (startapp "c:\\cid\\calfor")
  (dibtot nv lap lan)
  (pgosm)
)

(defun c:forj () (forj))


;*****************************************************************************
;* * *    SALVA LOS DATOS DE FORJADO PARA UTILIZARLOS EN EL CALCULO
;*****************************************************************************

(defun salva (nv lluz lg lq lan   lp la exi exd fckk fykk  b h b0 h0 sim
              dmin / fich n)

  (setq fich (open "c:\\forj.ddd" "w"))
  (print nv fich)
  (print b fich)
  (print h fich)
  (print b0 fich)
  (print h0 fich)
  (print fckk fich)
  (print fykk fich)
  (print gf fich)
  (print sim fich)
  (setq n 0)
  (repeat nv
    (setq n (1+ n))
    (print (nth n lluz) fich)
  )
  (print (nth 0 lan) fich)
  (setq n 0)
  (repeat nv
    (setq n (1+ n))
    (print (nth n lan) fich)
  )
  (setq n 0)
  (repeat nv
    (setq n (1+ n))
    (print (nth n lg) fich)
  )
  (setq n 0)
  (repeat nv
    (setq n (1+ n))
    (print (nth n lq) fich)
  )
  (setq n 0)
  (repeat nv
    (setq n (1+ n))
    (print (nth n lp) fich)
  )
  (setq n 0)
  (repeat nv
    (setq n (1+ n))
    (print (nth n la) fich)
  )
  (print exi fich)
  (print exd fich)
  (print dmin fich)
  (close fich)
)
```

```
;*****************************************************************************
;* * *  Dibuja manualmente un negativo
;*****************************************************************************
;

(defun neg (/ pp ang ar ar1 ar2 fi arm li ld a i d o qq r s tlin)

  (setq  pp  (getpoint "\nPunto de insercion")
         ang (getorient p "\nOrientacion de la armadura ")
         ar  (getstring "\nN§ barras <2>")
         fi  (getstring "\n%%C armaduras <10>")
         li  (getint "\nLongitud izquierda <100 cm>")
         ld  (getint "\nLongitud derecha <100 cm>")
  )
  (setq tlin (strcat "_FI" fi))
  (if (/= ar "")
    (setq ar1 ar)
    (setq ar1 "2")
  )
  (if (/= fi "")
    (setq ar2 fi)
    (setq ar2 "10")
  )
  (if (/= li nil)
    (setq i  (/ li 100.0)
          li (itoa li)
    )
    (setq i  1
          li "100"
    )
  )
  (if (/= ld nil)
    (setq d  (/ ld 100.0)
          ld (itoa ld)
    )
    (setq d  1
          ld "100"
    )
  )
  (setq  arm (strcat li " -  " ar1 "%%c" ar2 " - " ld)
         qq  (mapcar '+ pp (list (* d (cos ang)) (* d (sin ang))))
         o   (mapcar '- pp (list (* i (cos ang)) (* i (sin ang))))
         r   (mapcar '+
                     o
                     (list (* 0.05 (sin ang)) (* 0.05 (- (cos ang))))
             )
         s   (mapcar '+
                     qq
                     (list (* 0.05 (sin ang)) (* 0.05 (- (cos ang))))
             )
         ang (* ang (/ 180 3.1415))
  )
  (command "_LAYER" "_S" "negat" "")
  (command "_CHANGE" (entlast) "" "_P" "_LT" tlin "")
)

(defun c:neg () (neg))


;*****************************************************************************
;* * *  DIBUJA LAS ARMADURAS DE NEGATIVO
;
;       pa              punto del apoyo
;       ang             angulo de orientacion de la armadura en radianes
;       fi              diametro armadura
;       ar              posicion de armadura 1-2 o 3
```

```
;       li              longitud izq. mt.es cadena texto
;       ld              longitud derecha mt.cadena texto
;       izq             0 = prolonga       1 = patilla
;       der             0 = prolonga       1 = patilla
;       p               punto de insercion del negativo
;       pat             patilla doblada en metros
;       patt            patilla en texto
;*****************************************************************************
'
(defun dibneg (pa   ang   fi   ar   li    ld   izq  der  /    p     p1
               ar1  ar2   arm  fid  a     i    d    o    q    r     s
               r1   s1    ang1 pat  patt  tlin ult
              )


  (if (<= 15 h)
    (setq pat  0.10
          patt "10"
    )
  )
  (if (<= 20 h)
    (setq pat  0.15
          patt "15"
    )
  )
  (if (<= 26 h)
    (setq pat  0.20
          patt "20"
    )
  )
  (if (<= 30 h)
    (setq pat  0.25
          patt "25"
    )
  )
  (if (<= 35 h)
    (setq pat  0.30
          patt "30"
    )
  )
  (if (<= 40 h)
    (setq pat  0.35
          patt "35"
    )
  )
  (if (<= 45 h)
    (setq pat  0.40
          patt "40"
    )
  )
  (if (<= 50 h)
    (setq pat  0.45
          patt "45"
    )
  )
  (command "_LAYER" "_S" "textarm" "")

  (if (= ar 1)
    (setq p (polar pa (- ang 1.57075) 0.15))
  )
  (if (= ar 2)
    (setq p (polar pa (+ ang 1.57075) 0.25))
  )
  (if (= ar 3)
    (setq p pa)
  )
```

```
  (setq i    (/ (atoi li) 100.0)
        d    (/ (atoi ld) 100.0)
        fid  (/ (atof fi) 1000.0)
        tlin (strcat "_FI" fi)
        arm  (strcat li "- 1%%c" fi "- " ld)
        q    (mapcar '+ p (list (* d (cos ang)) (* d (sin ang))))
        o    (mapcar '- p (list (* i (cos ang)) (* i (sin ang))))
        r    (mapcar '+
                     o
                     (list (* 0.05 (sin ang)) (* 0.05 (- (cos ang))))
             )
        s    (mapcar '+
                     q
                     (list (* 0.05 (sin ang)) (* 0.05 (- (cos ang))))
             )
  )
  (if (= izq 0)
    (setq r1 r)
    (progn
      (setq r1    (polar r (- ang 1.57075) pat)
            p1    (polar r1 (- ang 3.141592) 0.15)
            ang1 (* ang (/ 180 3.1415))
      )
      (command "_TEXT" "_C" p1 ang1 patt)
    )
  )
  (if (= der 0)
    (setq s1 s)
    (progn
      (setq s1    (polar s (- ang 1.57075) pat)
            p1    (polar s1 ang 0.15)
            ang1 (* ang (/ 180 3.1415))
      )
      (command "_TEXT" "_C" p1 ang1 patt)
    )
  )
  (setq ang (* ang (/ 180 3.1415)))
  (command "_TEXT" "_C" p ang arm)
  (command "_LAYER" "_S" "negat" "")
  (command "_3DPOLY" r1 r s s1 "")
  (command "_CHANGE" (entlast) "" "_P" "_LT" tlin "")

  (setq ang (distance r1 r)
        ang (+ ang (distance r s))
        ang (+ ang (distance s s1))
  )
)


;*****************************************************************************
;* * *    LLAMA AL FICHERO DE RESULTADOS Y LO DIBUJA TODO
;*****************************************************************************

(defun dibtot (nv   lap   lan  /    p    p1   p2   ang  ang1 fi   ar
               li   ld    fich v    md   izq  der  vdi  vdd  an   an1
               an2  as1   fip  fcd  fyd  mc
              )

  (cprev)
  (noecho)
  (cposm)
  (cpscp)
  (cpcap)
  (r_non)
```

```
(setq  izq 1.0
       der 0.0
)
(if (= exi "P")
  (setq izq 0.0)
)
(setq  fich (open "c:\\forj.rrr" "r")
       p1   (nth 0 lap)
       p2   (nth 1 lap)
       ang  (angle p1 p2)
       ar   0
)
(repeat 3
  (setq ar   (1+ ar)
        fi   (read-line fich)
        li   (read-line fich)
        ld   (read-line fich)
        ang  (angle p1 p2)
        ang1 ang
  )
  (if (/= fi "0")
    (dibneg p1 ang1 fi ar li ld izq der)
  )
)
(setq  v   0
       izq 0.0
       der 0.0
)
(repeat nv
  (setq v (1+ v))
  (if (= v nv)
    (setq izq  0.0
          der  1.0
    )
    (setq izq  0.0
          der  0.0
    )
  )
  (if (= exd "V")
    (if (= v (1- nv))
      (setq der 1.0)
    )
  )
  (if (= exi "V")
    (if (= v 1)
      (setq izq 1.0)
    )
  )
  (setq p2  (nth v lap)
        ang (angle p1 p2)
        ar  0
  )
  (repeat 3
    (if (= v nv)
      (if (= exd "P")
        (setq der 0.0)
        (setq der 1.0)
      )
    )
    (setq ar    (1+ ar)
          fi    (read-line fich)
          li    (read-line fich)
          ld    (read-line fich)
          ang   (angle p1 p2)
          ang1 ang
    )
```

```
    (if (/= fi "0")
      (dibneg p2 ang1 fi ar li ld izq der)
    )
    (setq der  0.0
          izq  0.0
    )
  )
  (setq p1 p2)
)

(command "_LAYER" "_S" "textarm" "")
(setq p1 (nth 0 lap)
      v  0
)
(repeat nv
  (setq v  (1+ v)
        p2 (nth v lap)
        ar (read-line fich)
  )
  (if (/= ar ",00")
    (progn
      (setq ang  (angle p1 p2)
            p    (pmig p1 p2)
            ang1 (polar p (- ang 1.57075) 0.42)
            md   (strcat "Md=" ar " kNm/m")
            ang  (* ang (/ 180 3.1415))
      )
      (command "_TEXT" "_C" ang1 ang md)
    )
  )
  (if (= tipfor "In situ")
    (progn
      (setq fcd (* 10 (/ fck 1.5))
            fyd (* 10 (/ fyk 1.15))
      )
      (if (/= ar ",00")
        (setq md (* (atof ar) b0))
        (setq md 100)
      )
      (setq as1 (armat b h b0 h0 (+ rec 0.6) md fcd fyd)

            fip (buscafip as1)          ; busca fi positivos
            as1 (ponpos p1 p2 fip)
      )                                           ; dibuja positivos
    )
  )
  (setq p1 p2)
)
(setq p1  (nth 0 lap)
      an1 (/ (nth 0 lan) 200)
      v   0
)
(repeat nv
  (setq v   (1+ v)
        p2  (nth v lap)
        an2 (/ (nth v lan) 200)
        vdi (read-line fich)
        vdd (read-line fich)
  )
  (if (/= vdi "0")
    (progn
      (setq an   (+ an1 (/ (atof vdi) 100))
            ang  (angle p1 p2)
            p    (polar p1 ang an)
            ang1 (polar p (- ang 1.57075) 1.3)
            p    (polar ang1 (+ ang 1.57075) 2.6)
```

```
      )
      (command "_PLINE" ang1 p "")
      (setq p   (polar ang1 ang 0.1)
            md  (strcat "Macizado " vdi " cm")
            ang (* ang (/ 180 3.1415))
            ang (+ ang 90)
      )
      (command "_TEXT" "_C" p ang md)
    )
  )
  (if (/= vdd "0")
    (progn
      (setq an   (+ an2 (/ (atof vdd) 100))
            ang  (angle p2 p1)
            p    (polar p2 ang an)
            ang1 (polar p (- ang 1.57075) 1.2)
            p    (polar ang1 (+ ang 1.57075) 2.4)
      )
      (command "_PLINE" ang1 p "")
      (setq p   (polar ang1 ang 0.1)
            md  (strcat "Macizado " vdd " cm")
            ang (* ang (/ 180 3.1415))
            ang (+ ang 90)
      )
      (command "_TEXT" "_C" p ang md)
    )
  )
  (setq p1  p2
        an1 an2
  )
)

(setq p1 (nth 0 lap)                    ;------  coloca cortante
      v  0
)
(repeat nv
  (setq v  (1+ v)
        p2 (nth v lap)
        ar (read-line fich)
  )
  (if (/= ar ",00")
    (progn
      (setq ang  (angle p1 p2)
            ang1 (polar p1 (- ang 1.57075) 0.6)
            md   (strcat " Vd=" ar "kN ")
            ang  (* ang (/ 180 3.1415))
      )
      (command "_TEXT" "_BL" ang1 ang md)
    )
  )
  (setq ar (read-line fich))
  (if (/= ar "0")
    (progn
      (setq ang  (angle p1 p2)
            ang1 (polar p2 (- ang 1.57075) 0.6)
            md   (strcat " Vd=" ar "kN ")
            ang  (* ang (/ 180 3.1415))
      )
      (command "_TEXT" "_BR" ang1 ang md)
    )
  )
  (setq p1 p2)
)
                                        ;-----------------------
(close fich)
```

```
  (avisoUNDO)
  (pgosm)
  (pgscp)
  (pgcap)
)

;*****************************************************************************
;* * *  FUNCION PARA colocar  LOS POSITIVOS
;       p1 y p2 puntos extremos   fip texto de armadura    v par o impar
;*****************************************************************************

(defun ponpos (p1 pf fip / ang p pa lb lbb arm d12)

                                        ;  p1 y pf  puntos  de apoyos a eje
                                        ;  ang      angulo de orientacion de
                                        ;            la armadura en radianes
                                        ;  fip      diametro armadura

  (setq  pa  (pmig p1 pf)
         d12 (* 100 (+ (distance p1 pf) 0.1))
         ang (angle p1 pf)
         lb  (ajusta5 d12)
         lbb (itoa (fix lb))
         arm (strcat "2%%c" (itoa fip) "(" lbb ")")
  )
  (command "_LAYER" "_S" "posit" "")
  (setq  p   (polar pa (+ ang 1.57075) 0.05)
         ang (* ang (/ 180 3.1415))
  )
  (command "_TEXT" "C" p ang arm)
)


; *********      FUNCIONES ARMAR NERVIOS EN T DE HORMIGON ARMADO "in situ"


; armat          OBTIENE LA ARMADURA DE NERVIOS EN   T
; uss1           Devuelve US1 para  md < mlim
; uss2           Devuelve US2 para md > mlim
; buscafip       Devuelve diametro FI(mm) DE POSITIVOS NERVIOS
; ajusta5        REDONDEA A MULTIPLOS DE 5 cm
; insforja       INSERTA BLOQUES TIPO DE FORJADO
; editforja     EDITA LAS CARACTERISTICAS DEL FORJADO
; selforja       SELECCIONA LAS CARACTERISTICAS DEL FORJADO
; lanclb         Determina la longitud basica de anclaje
;


(prompt "Cargando las utilidades de ARMADO NERVIOS EN T \n")


;*****************************************************************************

;* * *           OBTIENE LA ARMADURA DE NERVIOS EN   T
;
; b0             ancho nervio
; h              canto total
; b              ancho losa ( intereje nervios)
; h0             canto losa
; md             momento mkp
;
; devuelve el area (cm2) de armadura de positivos en el nervio
; si la armadura es   NEGATIVA = SECCION INSUFICIENTE
;
;*****************************************************************************
```

```
(defun armat (b0 h b h0 rec md fcd fyd / d mm m0 mdd utc uta)

  (setq d   (- h rec)
        utc (* 0.85 fcd b h0)
        uta (* (- b b0) 0.85 fcd h0)
        mm  (* md 100)
  )
  (cond ((<= h0 (* 0.5 d))
         (progn
           (setq m0 (* utc (- d (* 0.5 h0))))
           (if (> mm m0)
             (setq mdd -1)
             (setq mdd mm)
           )
         )
        )
        ((> h0 (* 0.5 d))
         (progn
           (princ "\n Armadura2 h0 > 0.5d ")
         )
        )
  )
  (if (> mdd 0)
    (setq us1 (uss1 fcd b d mdd))
    (setq us1 -1000)
  )
  (setq as1 (/ us1 fyd))
)


;*****************************************************************************
;* * *           Devuelve US1 para  md < mlim
;
; b              ancho      cm
; d              canto util cm
; md             momento de calculo en kpxcm
;                devuelve Us1 en kp
;                u0 (* 0.85 fcd b d)
;
;*****************************************************************************

(defun uss1 (fcd b d md / u0 p)

  (setq u0 (* 0.85 fcd b d)
        p  (* u0 (- 1 (sqrt (- 1 (/ (* 2 md) u0 d)))))
  )
)


;*****************************************************************************
;* * *           Devuelve US2 para md > mlim
;
; b              ancho      cm
; d              canto util cm
; md             momento de calculo en kpxcm
;                devuelve Us2 en kp
;                u0 (* 0.85 fcd b d)
;
;*****************************************************************************

(defun uss2 (fcd b d rec md u0 / p d1)

  (setq d1 (- d rec)
        p  (/ (- md (* u0 d 0.375)) d1)
  )
```

```
)


;*****************************************************************************
;* * *           Devuelve diametro FI(mm) DE POSITIVOS NERVIOS
;
; as1            cm2 de armadura busca entre la serie preparada de redondos
; lar            lista de areas de 2fi
; lfi            lista de diametros fi en mm
;
;*****************************************************************************

(defun buscafip  (as1 / lar lfi ctrl n1 nv a1 a2 aa)

  (setq  lar '(0 1.01 1.57 2.26 4.02 6.28 9.82)
         lfi '(0 8 10 12 16 20 25)
         fi  0
         nv  0
  )
  (repeat 6
    (setq n1 nv
          nv (1+ nv)
          a1 (nth n1 lar)
          a2 (nth nv lar)
    )
    (if (> as1 a1)
      (progn
        (if (< as1 a2)
          (setq fi (nth nv lfi))
        )
      )
    )
  )
  (if (< fi dmin)
    (setq fi dmin)
  )
  (setq aa fi)
)


;*****************************************************************************
;* * *           REDONDEA A MULTIPLOS DE 5 cm
;
; cm             valor en cm
;
;*****************************************************************************

(defun ajusta5 (cm / rest x)

  (setq rest (rem cm 5))
  (if (> rest 0)
    (setq x (+ cm (- 5 rest)))
    (setq x cm)
  )
  (setq rest x)
)


;*****************************************************************************
;* * *   INSERTA BLOQUES TIPO DE FORJADO
;*****************************************************************************

(defun insforja  (tt / n1)

  (setq n1 (getpoint " Indicar punto de insercion"))
  (cond ((= tt 1) (command "_INSERT" "FUP_VS" n1 "" "" ""))
```

```
        ((= tt 2) (command "_INSERT" "FUA_VS" n1 "" "" ""))
        ((= tt 3) (command "_INSERT" "FUP_SS" n1 "" "" ""))
        ((= tt 4) (command "_INSERT" "FUA_SS" n1 "" "" ""))
        ((= tt 5) (command "_INSERT" "FUP_VD" n1 "" "" ""))
        ((= tt 6) (command "_INSERT" "FUA_VD" n1 "" "" ""))
        ((= tt 7) (command "_INSERT" "FUP_SD" n1 "" "" ""))
        ((= tt 8) (command "_INSERT" "FUA_SD" n1 "" "" ""))
        ((= tt 9) (command "_INSERT" "FUI_" n1 "" "" ""))
  )
)


;*****************************************************************************
;* * *     EDITA LAS CARACTERISTICAS DEL FORJADO
;*****************************************************************************

(defun editforja (/ ent n0 n1)

  (setq  ent (entsel "\nSeleccione el tipo de forjado: ")
         n0  (car ent)
         n1  (entget n0)
  )
  (if (= "INSERT" (cdr (assoc 0 n1)))
    (command "_DDATTE" n0)
  )
)


;*****************************************************************************
;* * *    SELECCIONA LAS CARACTERISTICAS DEL FORJADO
;*****************************************************************************

(defun selforja  (/ ent n0 n1 nfor tf ent1 nat vat ctrl)

  (setq  ent (entsel "\nSeleccione el tipo de forjado: ")
         n0  (car ent)
         n1  (entget n0)
  )
  (if (= "INSERT" (cdr (assoc 0 n1)))
    (setq nfor (cdr (assoc 2 n1)))
  )
  (if (= (substr nfor 1 2) "FU")
    (progn (if (= (substr nfor 3 1) "A")
             (setq tipfor "Viguetas Armadas")
           )
           (if (= (substr nfor 3 1) "P")
             (setq tipfor "Viguetas Pretens")
           )
           (if (= (substr nfor 3 1) "I")
             (setq tipfor "In situ")
           )
           (setq ctrl T)
           (while ctrl
             (setq ent1  (entnext n0)
                   n1   (entget ent1)
             )
             (if (= "ATTRIB" (cdr (assoc 0 n1)))
               (progn
                  (setq nat (cdr (assoc 2 n1))
                        vat (cdr (assoc 1 n1))
                  )
               )
             )
             (if (= "LE" nat)
               (setq fyk (atoi vat))
             )
```

```
            (if (= "RC" nat)
              (setq fck (atoi vat))
            )
            (if (= "PP" nat)
              (setq g (atof vat))
            )
            (if (= "SU" nat)
              (setq q (atof vat))
            )
            (if (= "I" nat)
              (setq b0 (atoi vat))
            )
            (if (= "H" nat)
              (setq h (atoi vat))
            )
            (if (= "V" nat)
              (setq b (atoi vat))
            )
            (if (= "L" nat)
              (setq h0 (atoi vat))
            )
            (if (= "DMIN" nat)
              (setq dmin (atoi vat))
            )
            (if (= "REC" nat)
              (setq rec (atof vat))
            )
            (if (= "SEQEND" (cdr (assoc 0 n1)))
              (setq ctrl nil)
            )
            (setq n0 ent1)
          )

    )
  )
)


;*****************************************************************************
;* * *          Determina la longitud basica de anclaje
;
; fi             diametro en mm de la barra
; pos            posicio 1 bajo      2 arriba
;                devuelve la long de anclaje en cm redondeada a 5
;
;*****************************************************************************

(defun lanclb (fi fyk fck pos / m m1 lb)

  (if (>= fyk 4300)
    (progn
      (setq m 16)
      (if (= fck 250)
        (setq m 15)
      )
      (if (= fck 300)
        (setq m 13)
      )
      (if (= fck 350)
        (setq m 12)
      )
      (if (= fck 400)
        (setq m 11)
      )
    )
    (progn
```

```
      (setq m 14)
      (if (= fck 250)
        (setq m 12)
      )
      (if (= fck 300)
        (setq m 10)
      )
      (if (= fck 350)
        (setq m 9)
      )
      (if (= fck 400)
        (setq m 8)
      )
    )
  )
  (if (= pos 2)
    (setq m (* m 1.4))
  )
  (setq  fi (/ fi 10)
         lb (* m fi fi)
         lb (ajusta5 lb)
  )
)
```